```
0000                    NAM     "Assist09 - MC6809 Monitor"

                ;*****************************************************************************
                ;*      This is the base ASSIST09 ROM. It may run with or without the        *
                ;*      extension ROM which when present will be automatically incorporated  *
                ;*      by the BLDVTR subroutine                                             *
                ;*      Assist09 source code published in:                                   *
                ;*      MC6809-MC6809E 8-Bit Microprocessor programming manual               *
                ;*      Page B-37 to B74,Section B.11 monitor listing                        *
                ;*      March 1, 1981, Reprinted May 1983 Motorola Inc.                      *
                ;*****************************************************************************

                ;*****************************************************************************
                ;*      Copyright (C) Motorola, Inc. 1979                                    *
                ;*****************************************************************************

                ;*** AVOCET XASM09 PSUDO-OPS ,CP/M environment Jul-1983
                ;FCB    OPSYN   DB              ;
                ;FCC    OPSYN   DB
                ;FDB    OPSYN   DW
                ;RMB    OPSYN   DS
                ;*** edited for asm6809 freeware 18-Feb-2025
                ; '$' changed back to '*'
                ; FCC psudo-op ASCII string entry changed from 'string' to /string/


                ;*****************************************************************************
                ;*      Global module equates                                                *
                ;*****************************************************************************

                ;*ROMBEG        EQU     $F400           ;* move forward for assy syntax debugging
F800            ROMBEG  EQU     $F800           ;* ROM STRAT ASSEMBLY ADDRESS
E700            RAMOFS  EQU     -$1900          ;* ROM OFFSET TO RAM WORK PAGE
0800            ROMSIZ  EQU     2048            ;* ROM SIZE
F000            ROM2OF  EQU     ROMBEG-ROMSIZ   ;* START OF EXTENSION ROM
E008            ACIA    EQU     $E008           ;* DEFAULT ACIA ADDRESS
E000            PTM     EQU     $E000           ;* DEFAULT PTM ADDRESS
0000            DFTCHP  EQU     0               ;* DEFAULT CHARACTER PAD COUNT
0005            DFTNLP  EQU     5
003E            PROMPT  EQU     '>'             ;* PROMPT CHARACTER
0008            NUMBKP  EQU     8               ;* NUMBER OF BREAK POINTS

                ;*****************************************************************************
                ;*      Miscelaneous equates                                                 *
                ;*****************************************************************************
0004            EOT     EQU     $04             ;* END OF TRANSMISSION
0007            BELL    EQU     $07             ;* BELL CHARACTER'
000A            LF      EQU     $0A             ;* LINEFEED
000D            CR      EQU     $0D             ;* CARRIAGE RETURN
0010            DLE     EQU     $10             ;* DATA LINK ESCAPE
0018            CAN     EQU     $18             ;* CANCEL (^X)
                ;*      PTM ACCESS DEFINITION
E001            PTMSTA  EQU     PTM+1           ;* READ STATUS REGISTER
E000            PTMC13  EQU     PTM             ;* CONTROL REGISTER 1 AND 3
E001            PTMC2   EQU     PTM+1           ;* CONTROL REGISTER 2
E002            PTMTM1  EQU     PTM+2           ;* LATCH 1
E004            PTMTM2  EQU     PTM+4           ;* LATCH 2
E006            PTMTM3  EQU     PTM+6           ;* LATCH 3

008C            SKIP2   EQU     $8C             ;* "CMPX #" OPCODE - SKIP TWO BYTES

                ;*****************************************************************************
                ;*      The following equates define functions provided by the assist09      *
                ;*      monitor via the SWI instruction.                                     *
                ;*****************************************************************************

0000            INCHNP  EQU     0               ;* INPUT CHAR IN A REG - NOPARITY
0001            OUTCH   EQU     1               ;* OUTPUT CHAR FROM A REG
0002            PDATA1  EQU     2               ;* OUTPUT STRING
0003            PDATA   EQU     3               ;* OUTPUT CR/LF THEN STRING
0004            OUT2HS  EQU     4               ;* OUTPUT TWO HEX AND SPACE
0005            OUT4HS  EQU     5               ;* OUTPUT FOUR HEX AND SPACE
0006            PCRLF   EQU     6               ;* OUTPUT CR/LF
0007            SPACE   EQU     7               ;* OUTPUT A SPACE
0008            MONITR  EQU     8               ;* ENTER ASSIST09 MONITOR
0009            VCTRSW  EQU     9               ;* VECTOR EXAMINE/SWITCH
000A            BRKPT   EQU     10              ;* USER PROGRAM BREAKPOINT
000B            PAUSE   EQU     11              ;* TASK PAUSE FUNCTION
000B            NUMFUN  EQU     11              ;* NUMBER OF AVAILABLE FUNCTIONS
                ;*
                ;* Next sub-codes for accessing the vector table.
                ;* They are equivalent to offsets in the table.
                ;* relative positioning must be maintained.
                ;*
0000            _AVTBL  EQU     0               ;* ADDRESS OF VECTOR TABLE
0002            _CMDL1  EQU     2               ;* FIRST COMMAND LIST
0004            _RSVD   EQU     4               ;* RESERVED HARDWARE VECTOR
0006            _SWI3   EQU     6               ;* SWI3 SUBROUTINE
```

```
0008                _SWI2   EQU     8               ;* SWI2 SUBROUTINE
000A                _FIRQ   EQU     10              ;* FIRQ SUBROUTINE
000C                _IRQ    EQU     12              ;* IRQ ROUTINE
000E                _SWI    EQU     14              ;* SWI ROUTINE
0010                _NMI    EQU     16              ;* NMI ROUTINE
0012                _RESET  EQU     18              ;* RESET ROUTINE
0014                _CION   EQU     20              ;* CONSOLE ON
0016                _CIDTA  EQU     22              ;* CONSOLE INPUT DATA
0018                _CIOFF  EQU     24              ;* CONSOLE INPUT OFF
001A                _COON   EQU     26              ;* CONSOLE OUTPUT ON
001C                _CODTA  EQU     28              ;* CONSOLE OUTPUT DATA
001E                _COOFF  EQU     30              ;* CONSOLE OUTPUT OFF
0020                _HSDTA  EQU     32              ;* HIGH SPEED PRINTDATA
0022                _BSON   EQU     34              ;* PUNCH/LOAD ON
0024                _BSDTA  EQU     36              ;* PUNCH/LOAD DATA
0026                _BSOFF  EQU     38              ;* PUNCH/LOAD OFF
0028                _PAUSE  EQU     40              ;* TASK PAUSE ROUTINE
002A                _EXPAN  EQU     42              ;* EXPRESSION ANALYZER
002C                _CMDL2  EQU     44              ;* SECOND COMMAND LIST
002E                _ACIA   EQU     46              ;* ACIA ADDRESS
0030                _PAD    EQU     48              ;* CHARACTER PAD AND NEW LINE PAD
0032                _ECHO   EQU     50              ;* ECHO/LOAD AND NULL BKPT FLAG
0034                _PTM    EQU     52              ;* PTM ADDRESS
001B                NUMVTR  EQU     52/2+1          ;* NUMBER OF VECTORS
0034                HIVTR   EQU     52              ;* HIGHEST VECTOR OFFSET

                    ;*******************************************************************************
                    ;*                              WORK AREA                                      *
                    ;*      This work area is assigned to the page addressed by -$1800, PCR        *
                    ;*      from the base address of the ASSIST09 ROM. The direct page register    *
                    ;*      during operations will point this work area. The stack initially       *
                    ;*      starts under the reserved work areas as defined herin.                 *
                    ;*******************************************************************************

DF00                WORKPAG EQU     ROMBEG+RAMOFS   ;* SETUP DIRECT PAGE ADDRESS
0000                        SETDP   WORKPAG >> 8    ;* NOTIFY ASSEMBLER
E000                        ORG     WORKPAG+256     ;* READY PAGE DEFINITION
                    ;*      The following thru BKPTOP must reside in this order
                    ;*      For proper initialization


DFFC                        ORG     *-4
DFFC                PAUSER  EQU     *               ;* PAUSE ROUTINE
DFFB                        ORG     *-1
DFFB                SWIBFL  EQU     *               ;* BYPASS SWI AS BREAKPOINT FLAG
DFFA                        ORG     *-1
DFFA                BKPTCT  EQU     *               ;* BREAKPOINT COUNT
DFF8                        ORG     *-2
DFF8                SLEVEL  EQU     *               ;* STACK TRACE LEVEL
DFC2                        ORG     *-NUMVTR*2
DFC2                VECTAB  EQU     *               ;* VECTOR TABLE
DFB2                        ORG     *-2*NUMBKP
DFB2                BKPTBL  EQU     *               ;* BREAKPOINT TABLE
DFA2                        ORG     *-2*NUMBKP
DFA2                BKPTOP  EQU     *               ;* BREAKPOINT OPCODE TABLE
DFA0                        ORG     *-2
DFA0                WINDOW  EQU     *               ;* WONDOW
DF9E                        ORG     *-2
DF9E                ADDR    EQU     *               ;* ADDRESS POINTER VALUE
DF9D                        ORG     *-1
DF9D                BASEPG  EQU     *               ;* BASE PAGE VALUE
DF9B                        ORG     *-2
DF9B                NUMBER  EQU     *               ;* BINARY BUILD AREA
DF99                        ORG     *-2
DF99                LASTOP  EQU     *               ;* LAST OPCODE TRACED
DF97                        ORG     *-2
DF97                RSTACK  EQU     *               ;* RESET STAK POINTER
DF95                        ORG     *-2
DF95                PSTACK  EQU     *               ;* COMMAND RECOVERY STACK
DF93                        ORG     *-2
DF93                PCNTER  EQU     *               ;* LAST PROGRAM COUNTER
DF91                        ORG     *-2
DF91                TRACEC  EQU     *               ;* TRACE COUNT
DF90                        ORG     *-1
DF90                SWICNT  EQU     *               ;* TRACE "SWI" NEST LEVEL COUNT
DF8F                        ORG     *-1
DF8F                MISFLG  EQU     *               ;* LOAD CMD/THRU BREAKPOINT FLAG
DF8E                        ORG     *-1
DF8E                DELIM   EQU     *               ;* EXPRESSION DELIMITER/WORK AREA
DF66                        ORG     *-40
DF66                ROM2WK  EQU     *               ;* EXPRESSION ROM RESERVER AREA
DF51                        ORG     *-21
DF51                TSTACK  EQU     *               ;* TEMPORARY STACK HOLD
DF51                STACK   EQU     *               ;* START OF INITIAL STACK

                    ;*******************************************************************************
                    ;*      Default the ROM beginning address to 'ROMBEG' ASSIST09 is position     *
                    ;*      address dependent , however we assemble assuming control of the        *
```

```
                        ;*      hardware vectors. Note that the work RAM page must be 'RAMOFS' from   *
                        ;*      the ROM beginning address.                                            *
                        ;*****************************************************************************

F800                            ORG     ROMBEG          ;* ROM ASSEMBLY/DFAULT ADDRESS

                        ;*****************************************************************************
                        ;*      BLDVAR  -       BUILD ASSIST09 VECTOR TABLE                           *
                        ;*      HARDWARE RESET CALL THIS SUBROUTINE TO BULID THE ASSIST09 VECTOR       *
                        ;*      TABLE. THIS SUBROUTINE RESIDES AT THE FIRST BYTE OF THE ASSIST09       *
                        ;*      ROM, AND CAN BE CALLED VIA EXTERNAL CONTROL CODE FOR REMOTE ASSIST09   *
                        ;*      EXECUTION.                                                            *
                        ;*      INPUT:  S       -       VALID STACK RAM                               *
                        ;*      OUTPUT: U       -       VECTOR TABLE ADDRESS                          *
                        ;*              DPR     -       ASSIST09 WORK AREA PAGE                       *
                        ;*      THE VECTOR TABLE AND DEFAULTS ARE INITIALIZED                         *
                        ;*      ALL REGISTERS VOLATILE                                                *
                        ;*****************************************************************************

F800                    BLDVTR
F800  308DE7BE                  LEAX    VECTAB,PCR      ;* ADDRESS VECTOR TABLE
F804  1F10                      TFR     X,D             ;* OBTAIN BASE PAGE ADDRESS
F806  1F8B                      TFR     A,DP            ;* SETUP DPR
F808  979D                      STA     BASEPG          ;* STORE FOR QUICK REFERENCE
F80A  3384                      LEAU    ,X              ;* RETURN TABLE TO CALLER
F80C  318C35                    LEAY    <INITVT,PCR     ;* LOAD FROM ADDR
F80F  EF81                      STU     ,X++            ;* INIT VECTOR TABLE ADDRESS
F811  C616                      LDB     #NUMVTR-5       ;* NUMBER RELOCATABLE VETCORS
F813  3404                      PSHS    B               ;* STORE INDEX ON STACK
F815                    BLD2
F815  1F20                      TFR     Y,D             ;* PREPARE ADDRESS RESOLVE
F817  E3A1                      ADDD    ,Y++            ;* TO ABSOLUTE ADDRESS
F819  ED81                      STD     ,X++            ;* INTO VECTOR TABLE
F81B  6AE4                      DEC     ,S              ;* COUTN DOWN
F81D  26F6                      BNE     BLD2            ;* BRANCH IF MORE TO INSERT
F81F  C60D                      LDB     #INTVE-INTVS    ;* STATIC VALUE INIT LENGTH
F821                    BLD3
F821  A6A0                      LDA     ,Y+             ;* LOAD NEXT BYTE
F823  A780                      STA     ,X+             ;* STORE INTO POSITION
F825  5A                        DECB                    ;* COUNT DOWN
F826  26F9                      BNE     BLD3            ;* LOOP UNTIL DONE
F828  318DF7D4                  LEAY    ROM2OF,PCR      ;* TEST POSSIBLE EXTENSION ROM
F82C  8E20FE                    LDX     #$20FE          ;* LOAD "BRA *" FLAG PATTERN
F82F  ACA1                      CMPX    ,Y++            ;* ? EXTENDED ROM HERE
F831  2602                      BNE     BLDRTN          ;* BRANCH NOT OUR ROM TO RETURN
F833  ADA4                      JSR     ,Y              ;* CALL EXTENDED ROM INITIALIZE
F835  3584              BLDRTN  PULS    PC,B            ;* RETURN TO INITIALIZER

                        ;*****************************************************************************
                        ;*                              RESET ENTRY POINT                            *
                        ;*      HARDWARE RESET ENTERS HERE IF ASSIST09 IS ENABLED TO RECEIVE THE      *
                        ;*      MC6809 HARDWARE VECTORS. WE CALL THE BLDVTR SUBROUTINE TO             *
                        ;*      INITIALIZE THE VECTOR TABLE, AND THEN FIREUP THE MONITOR VIA          *
                        ;*      SWI CALL.                                                             *
                        ;*****************************************************************************

F837                    RESET
F837  328DE716                  LEAS    STACK,PCR       ;* SETUP INITIAL STACK
F83B  8DC3                      BSR     BLDVTR          ;* BUILD VECTOR TABLE
F83D                    RESET2
F83D  4F                        CLRA                    ;* ISSUE STARTUP MASSAGE
F83E  1F8B                      TFR     A,DP            ;* DEFAULT TO PAGE ZERO
F840  3F                        SWI                     ;* PERFORM MONITOR FIREUP
F841  08                        FCB     MONITR          ;* TO ENTER COMMAND PRECESSING
F842  20F9                      BRA     RESET2          ;* REENTER MONITOR IF 'CONTINUE'

                        ;*****************************************************************************
                        ;*              INITVT  -       INITIAL VECTOR TABLE                          *
                        ;*      THIS TABLE IS RELOCATED TO RAM AND REPRESENTS THE INITIAL STATE       *
                        ;*      OF THE VECTOR TABLE. ALL ADDRESSES ARE CONVERTED TO ABSOLUTE FORM.    *
                        ;*      THIS TABLE STARTS WITH THE SECOND ENTRY, ENDS WITH STATIC CONSTANT    *
                        ;*      INITIALIZATION DATA WHICH CARRIES BEYOND THE TABLE.                   *
                        ;*****************************************************************************

F844  0158              INITVT  FDB     CMDTBL-*        ;* DEFAULT FIRST COMMAND TABLE
F846  0292                      FDB     RSRVDR-*        ;* DEFAULT UNDEFINED HARDWARE VECTOR
F848  0290                      FDB     SWI3R-*         ;* DEFAULT SWI3
F84A  028E                      FDB     SWI2R-*         ;* DEFAULT SWI2
F84C  0270                      FDB     FIRQR-*         ;* DEFAULT FIRQ
F84E  028A                      FDB     IRQR-*          ;* DEFAULT IRQ ROUTINE
F850  0045                      FDB     SWIR-*          ;* DEFAULT SWI ROUTINE
F852  022B                      FDB     NMIR-*          ;* DEFAULT NMI ROUTINE
F854  FFE3                      FDB     RESET-*         ;* RESTART VECCTOR
F856  0290                      FDB     CION-*          ;* DEFAULT CION
F858  0284                      FDB     CIDTA-*         ;* DEFAULT CIDTA
F85A  0296                      FDB     CIOFF-*         ;* DEFAULT CIOFF
F85C  028A                      FDB     COON-*          ;* DEFAULT COON
F85E  0293                      FDB     CODTA-*         ;* DEFAULT CODTA
```

```
F860  0290                    FDB     COOFF-*         ;* DEFAULT COOFF
F862  039A                    FDB     HSDTA-*         ;* DEFAULT HSDTA
F864  02B7                    FDB     BSON-*          ;* DEFAULT BSON
F866  02D2                    FDB     BSDTA-*         ;* DEFAULT BSDTA
F868  02BF                    FDB     BSOFF-*         ;* DEFAULT BOFF
F86A  E792                    FDB     PAUSER-*        ;* DEFAULT PAUSER ROUTINE
F86C  047D                    FDB     EXP1-*          ;* DEFAULT EXPRESSION ANALYZER
F86E  012D                    FDB     CMDTB2-*        ;* DEFAULT SECODE COMMAND TABLE
                      ;*      CONSTANTS
F870                  INTVS
F870  E008                    FDB     ACIA            ;* DEFAULT ACIA
F872  0005                    FCB     DFTCHP,DFTNLP   ;* DEFAULT NULL PADDS
F874  0000                    FDB     0               ;* DEFAULT ECHO
F876  E000                    FDB     PTM             ;* DEFAULT PTM
F878  0000                    FDB     0               ;* INITIAL STACK TRACE LEVEL
F87A  00                      FCB     0               ;* INITIAL BREAKPOINT COUNT
F87B  00                      FCB     0               ;* SWI BREAKPOINT LEVEL
F87C  39                      FCB     $39             ;* DEFAULT PAUSE ROUTINE (RTS)
F87D                  INTVE   EQU     *
                      ;* B

                      ;*******************************************************************************
                      ;*                       ASSIST09 SWI HANDLER                                  *
                      ;*      The SWI handler provides all interfacing necessary for a user          *
                      ;*      program. A function byte is assumed to follow the SWI instruction.     *
                      ;*      it is bound checked and the proper routine is given control. This      *
                      ;*      invocation may also be a breakpoint interrupt. If so, the              *
                      ;*      breakpoint handler is entered.                                         *
                      ;*                                                                             *
                      ;*      Input : Machine state defined for SWI                                  *
                      ;*      Output: Varies according to function called. PC on callers stack       *
                      ;*              incremented by one if valid call.                              *
                      ;*      Volatile registers : see functions called                              *
                      ;*      State : Runs disabled unless function clears I flag.                   *
                      ;*******************************************************************************

                      ;*      SWI function vector table

F87D                  SWIVTB
F87D  0194                    FDB     ZINCH-SWIVTB    ;* INCHNP
F87F  01B1                    FDB     ZOTCH1-SWIVTB   ;* OUTCH
F881  01CB                    FDB     ZPDTA1-SWIVTB   ;* PDATA1
F883  01C3                    FDB     ZPDATA-SWIVTB   ;* PDATA
F885  0175                    FDB     ZOT2HS-SWIVTB   ;* OUT2HS
F887  0173                    FDB     ZOT4HS-SWIVTB   ;* OUT4HS
F889  01C0                    FDB     ZPCRLF-SWIVTB   ;* PCRLF
F88B  0179                    FDB     ZSPACE-SWIVTB   ;* SPACE
F88D  0055                    FDB     ZMONTR-SWIVTB   ;* MONITR
F88F  017D                    FDB     ZVSWTH-SWIVTB   ;* VCTRSW
F891  0256                    FDB     ZBKPNT-SWIVTB   ;* BREAKPOINT
F893  01D1                    FDB     ZPAUSE-SWIVTB   ;* TASK PAUSE

F895                  SWIR
F895  6A8DE6F7                LDEC    SWICNT,PCR      ;* UP "SWI" LEVEL FOR TRACE
F899  170225                  LBSR    LDDP            ;* SETUP PAGE AND VERIFY STACK
                      ;*      CHECK FOR BREAKPOINT TRAP
F89C  EE6A                    LDU     10,S            ;* LOAD PROGRAM COUNTER
F89E  335F                    LEAU    -1,U            ;* BACK TO SWI ADDRESS
F8A0  0DFB                    TST     SWIBFL          ;* ? THIS "SWI" BREAKPOINT
F8A2  2611                    BNE     SWIDNE          ;* BRANCH IF SO TO LET THROUGN
F8A4  17069B                  LBSR    CBKLDR          ;* OBTAIN BREAKPOINT POINTERS
F8A7  50                      NEGB                    ;* OBTAIN POSITIVE COUNT
F8A8                  SWILP
F8A8  5A                      DECB                    ;* COUNT DOWN
F8A9  2B0A                    BMI     SWIDNE          ;* BRANCH WHEN DONE
F8AB  11A3A1                  CMPU    ,Y++            ;* ? WAS THIS A BREAKPOINT
F8AE  26F8                    BNE     SWILP           ;* BRANCH IS NOT
F8B0  EF6A                    STU     10,S            ;* SET PROGRAM COUNTER BACK
F8B2  16021E                  LBRA    ZBKPNT          ;* GO TO BREAKPOINT
F8B5                  SWIDNE
F8B5  0FFB                    CLR     SWIBFL          ;* CLEAR IN CASE SET
F8B7  3706                    PULU    D               ;* OBTAIN FUNCTION BYTE, UP PC
F8B9  C10B                    CMPB    #NUMFUN         ;* ? TOO HIGH
F8BB  1022020F                LBHI    ERROR           ;* YES , DO BREAKPOINT
F8BF  EF6A                    STU     10,S            ;* BUMP PROGRAM COUNTER PAST SWI
F8C1  58                      ASLB                    ;* FUNCTION CODE TIMES TWO
F8C2  338CB8                  LEAU    SWIVTB,PCR      ;* OBTAIN VECTOR BRANCH ADDRESS
F8C5  ECC5                    LDD     B,U             ;* LOAD OFFSET
F8C7  6ECB                    JMP     D,U             ;* JUMP TO ROUTINE

                      ;*******************************************************************************
                      ;*              REGISTERS TO FUNCTION ROUTINE                                  *
                      ;*      DP      ->      work area page                                         *
                      ;*      D,Y,U   =       unreliable      X       =       as called from user    *
                      ;*      S       =       as from SWI interrupt                                  *
                      ;*******************************************************************************

                      ;*******************************************************************************
```

```
                    ;*                          [ SWI FUNCTION 8 ]                                 *
                    ;*                             MONITOR ENTRY                                   *
                    ;*      Fireup the ASSIST09 monitor.                                           *
                    ;*      The stack with its values for the direct page                          *
                    ;*      Register and condition code flags are used as is.                      *
                    ;*      1> initialize console I/O.                                             *
                    ;*      2> optional print signon.                                              *
                    ;*      3> initalize PTM for single stepping                                   *
                    ;*      4> Enter command processor                                             *
                    ;*                                                                             *
                    ;*      Input:  A=0 init console and print startup message                     *
                    ;*              A#0 omit console init and startup message                      *
                    ;*******************************************************************************

F8C9                SIGNON
F8C9  4153534953543039          FCC     "ASSIST09"      ;* SIGNON EYE-CATCHER
F8D1  04                        FCB     EOT
F8D2                ZMONTR
F8D2  10DF97                    STS     RSTACK          ;* SAVE FOR BAD STACK RECOVERY
F8D5  6D61                      TST     1,S             ;* ? INIT CONSOLE AND SEND MSG
F8D7  260D                      BNE     ZMONT2          ;* BRANCH IF NOT
F8D9  AD9DE6F9                  JSR     [VECTAB+_CION,PCR]      ;* READY CONSOLE INPUT
F8DD  AD9DE6FB                  JSR     [VECTAB+_COON,PCR]      ;* READY CONSOLE OUTPUT
F8E1  308CE5                    LEAX    SIGNON,PCR      ;* READY SIGNON EYE-CATCHER
F8E4  3F                        SWI                     ;* PERFORM
F8E5  03                        FCB     PDATA           ;* PRINT STRING
F8E6                ZMONT2
F8E6  9EF6                      LDX     VECTAB+_PTM     ;* LOAD PTM ADDRESS
F8E8  270D                      BEQ     CMD             ;* BRANCH IF NOT TO USE A PTM
F8EA  6F02                      CLR     PTMTM1-PTM,X    ;* SET LATCH TO CLEAR RESET
F8EC  6F03                      CLR     PTMTM1+1-PTM,X  ;* AND SET GATE HIGH
F8EE  CC01A6                    LDD     #$01A6          ;* SETUP TIMER 1 MODE
F8F1  A701                      STA     PTMC2-PTM,X     ;* SETUP FOR CONTROL REGISTER 1
F8F3  E784                      STB     PTMC13-PTM,X    ;* SET OUTPUT ENABLED/
                    ;*
                    ;*      SINGLE SHOT/ DUAL 8 BIT/INTERNAL MODE/ OPERATE
                    ;*
F8F5  6F01                      CLR     PTMC2-PTM,X     ;* SET CR2 BACK TO RESET FORM
                    ;*      FALL INTO COMMAND PROCESSOR

                    ;*******************************************************************************
                    ;*      COMMAND HANDLER                                                        *
                    ;*                                                                             *
                    ;*      Breakpoints are removed at this time.                                  *
                    ;*      Prompt for a command  and store all characters                         *
                    ;*      Until a separator on the stack.                                        *
                    ;*      Search for first matching command subset,                              *
                    ;*      Call it or give "?" response.                                          *
                    ;*      During command search:                                                 *
                    ;*      B       =       offset to next entry on X                              *
                    ;*      U       =       saved S                                                *
                    ;*      U-1     =       entry size                                             *
                    ;*      U-2     =       valid number flag (>=0 valid)/compare cnt              *
                    ;*      U-3     =       carriage return flag                                   *
                    ;*      U-4     =       start of command store                                 *
                    ;*      S+0     =       end of command store                                   *
                    ;*******************************************************************************

F8F7                CMD
F8F7  3F                        SWI                     ;* TO NEW LINE
F8F8  06                        FCB     PCRLF           ;* FUNCTION
                    ;*      DISARM THE BREAKPOINTS
F8F9                CMDNEP
F8F9  170646                    LBSR    CBKLDR          ;* OBTAIN BREAKPOINT POINTERS
F8FC  2A0C                      BPL     CMDNOL          ;* BRANCH IF NOT ARMED OR NONE
F8FE  50                        NEGB                    ;* MAKE POSITIVE
F8FF  D7FA                      STB     BKPTCT          ;* FLAG AS DISARMED
F901                CMDDDL
F901  5A                        DECB                    ;* FINISHED ?
F902  2B06                      BMI     CMDNOL          ;* BRANCH IF NOT
F904  A630                      LDA     -NUMBKP*2,Y     ;* LOAD OPCODE STORED
F906  A7B1                      STA     [,Y++]          ;* STORE BACK OVER "SWI"
F908  20F7                      BRA     CMDDDL          ;* LOOP UNTIL DONE
F90A                CMDNOL
F90A  AE6A                      LDX     10,S            ;* LOAD USERS PROGRAM COUNTER
F90C  9F93                      STX     PCNTER          ;* SAVE FOR EXPRESSION ANALYZER
F90E  863E                      LDA     #PROMPT         ;* LOAD PROMPT CHARACTER
F910  3F                        SWI                     ;* SEND TO OUTPUT HANDLER
F911  01                        FCB     OUTCH           ;* FUNCTION
F912  33E4                      LEAU    ,S              ;* REMEMBER STACK RESTORE ADDRESS
F914  DF95                      STU     PSTACK          ;* REMEMBER STACK FOR ERROR USE
F916  4F                        CLRA                    ;* PREPARE ZERO
F917  5F                        CLRB                    ;* PREPARE ZERO
F918  DD9B                      STD     NUMBER          ;* CLEAR NUMBER BUILD AREA
F91A  DD8F                      STD     MISFLG          ;* CLEAR MISCEL. AND SWICNT FLAGS
F91C  DD91                      STD     TRACEC          ;* CLEAR TRACE COUNT
F91E  C602                      LDB     #2              ;* SET D TO TWO
F920  3407                      PSHS    D,CC            ;* PLACE DEFAULTS ONTO STACK
```

```
                        ;*      CHECK FOR "QUICK" COMMANDS
F922  170454                    LBSR    READ            ;* OBTAIN FIRST CHARACTER
F925  308D0581                  LEAX    CDOT+2,PCR      ;* PRESET FOR SINGLE TRACE
F929  812E                      CMPA    #'.'            ;* QUICK TRACE ?
F92B  275A                      BEQ     CMDXQT          ;* BRANCH EQUAL FOR TRACE ONE
F92D  308D04E9                  LEAX    CMPADP+2,PCR    ;* READY MEMORY ENTRY POINT
F931  812F                      CMPA    #'/'            ;* OPEN LAST USED MEMORY ?
F933  2752                      BEQ     CMDXQT          ;* BRANCH TO DO IT IF SO
                        ;* PROCESS NEXT CHARACTER
F935                    CMD2
F935  8120                      CMPA    #' '            ;* BLANK OF DELIMITER ?
F937  2314                      BLS     CMDGOT          ;* BRANCH YES, WE HAVE IT
F939  3402                      PSHS    A               ;* BUILD ONTO STACK
F93B  6C5F                      INC     -1,U            ;* COUNT THIS CHARACTER
F93D  812F                      CMPA    #'/'            ;* MEMORY COMMAND ?
F93F  274F                      BEQ     CMDMEM          ;* BRANCH IF SO
F941  17040B                    LBSR    BLDHXC          ;* TREAT AS HEX VALUE
F944  2702                      BEQ     CMD3            ;* BRANCH IF STILL VALID NUMBER
F946  6A5E                      DEC     -2,U            ;* FLAG AS INVALID NUMBER
F948                    CMD3
F948  17042E                    LBSR    READ            ;* OBTAIN NEXT CHARACTER
F94B  20E8                      BRA     CMD2            ;* TEST NEXT CHARACTER
                        ;* GOT COMMAND, NOW SEARCH TABLES
F94D                    CMDGOT
F94D  800D                      SUBA    #CR             ;* SET ZERO IF CARRIAGE RETURN
F94F  A75D                      STA     -3,U            ;* SETUP FLAG
F951  9EC4                      LDX     VECTAB+_CMDL1   ;* START WITH FIRST CMD LIST
F953                    CMDSCH
F953  E680                      LDB     ,X+             ;* LOAD ENTRY LENGTH
F955  2A10                      BPL     CMDSME          ;* BRANCH ID NOT LIST END
F957  9EEE                      LDX     VECTAB+_CMDL2   ;* NOW TO SECOND COMMAND LIST
F959  5C                        INCB                    ;* TO CONTINUE TO DEFAULT LIST
F95A  27F7                      BEQ     CMDSCH          ;* BRAHCH IS SO
F95C                    CMDBAD
F95C  10DE95                    LDS     PSTACK          ;* RESTORE STACK
F95F  308D015A                  LEAX    ERRMSG,PCR      ;* POINT TO ERROR STRING
F963  3F                        SWI                     ;* SEND OUT
F964  02                        FCB     PDATA1          ;* TO CONSOLE
F965  2090                      BRA     CMD             ;* AND TRY AGAIN
                        ;* SEARCH NEXT ENTRY
F967                    CMDSME
F967  5A                        DECB                    ;* TAKE ACCOUNT OF LENGTH BYTE
F968  E15F                      CMPB    -1,U            ;* ENTERED LONGER THAN ENTRY ?
F96A  2403                      BHS     CMDSIZ          ;* BRANCH IF NOT TOO LONG
F96C                    CMDFLS
F96C  3A                        ABX                     ;* SKIP TO NEXT ENTRY
F96D  20E4                      BRA     CMDSCH          ;* AND TRY NEXT
F96F                    CMDSIZ
F96F  315D                      LEAY    -3,U            ;* PREPARE TO COMPARE
F971  A65F                      LDA     -1,U            ;* LOAD SIZE+2
F973  8002                      SUBA    #2              ;* TO ACTUAL SIZE ENTERED
F975  A75E                      STA     -2,U            ;* SAVE SIZE FOR COUNTDOWN
F977                    CMDCMP
F977  5A                        DECB                    ;* DOWN ONE BYTE
F978  A680                      LDA     ,X+             ;* NEXT COMMAND CHARACTER
F97A  A1A2                      CMPA    ,-Y             ;* SAME AS THAT ENTERED
F97C  26EE                      BNE     CMDFLS          ;* BRANCH IF TO FLUSH , IF NOT
F97E  6A5E                      DEC     -2,U            ;* COUNT DOWN LENGTH OF ENTRY
F980  26F5                      BNE     CMDCMP          ;* BRANCH IF MORE TO TEST
F982  3A                        ABX                     ;* TO NEXT ENTRY
F983  EC1E                      LDD     -2,X            ;* LOAD OFFSET
F985  308B                      LEAX    D,X             ;* COMPUTE ROUTINE ADDRESS+2
F987                    CMDXQT
F987  6D5D                      TST     -3,U            ;* SET CC FOR CARRIAGE RETURN TEST
F989  32C4                      LEAS    ,U              ;* DELETE STACK WORK AREA
F98B  AD1E                      JSR     -2,X            ;* CALL COMMAND
F98D  16FF7A                    LBRA    CMDNOL          ;* GO GET NEXT COMMAND
F990                    CMDMEM
F990  6D5E                      TST     -2,U            ;* VALID HEX NUMBER ENTERED
F992  2BC8                      BMI     CMDBAD          ;* BRANCH ERROR IF NOT
F994  3088AE                    LEAX    <CMEMN-CMPADP,X ;* TO DIFFERENT ENTRY
F997  DC9B                      LDD     NUMBER          ;* LOAD NUMBER ENTERED
F999  20EC                      BRA     CMDXQT          ;* AND ENTER MEMORY COMMAND

                        ;**     COMMANDS ARE ENTERED AS A SUBROUTINE WITH:
                        ;**     DPR-> ASSIST09 DIRECT PAGE WORK AREA
                        ;**     Z=1 CARRIAGE RETURN ENTERED
                        ;**     Z=0 NON CARRIAGE RETURN DELIMITER
                        ;**     S=NORMAL RETURN ADDRESS
                        ;** THE LABEL "CMDBAD" MAY BE ENTERED TO ISSUE AN ERROR FLAG

                        ;*******************************************************************************
                        ;*                      ASSSIST09 COMMAND TABLES                               *
                        ;*      THESE ARE THE DEFAULT COMMAND TABLES. EXTERNAL TABLES OF THE SAME      *
                        ;*      FORMAT MAY EXTEND/REPLACE THESE BY USING THE VECTOR SWAP FUNCTION.     *
                        ;*                                                                             *
                        ;*      Entry format:                                                          *
                        ;*      +0...Total size of entry (including this byte)                         *
```

```
                        ;*      +1...Command string                                             *
                        ;*      +N...two byte offset to command (ENTRYADDR-*)                   *
                        ;*                                                                      *
                        ;*      The table terminates with a one byte -1 or -2.                  *
                        ;*      The -1 continues the command search with the second command table     *
                        ;*      The -2 terminates command searches.                             *
                        ;**************************************************************************

                        ;* THIS IS THE DEFAULT LIST FOR THE SECOND COMMAND LIST ENTRY

F99B                    CMDTB2
F99B  FE                        FCB     -2              ;* STOP COMMAND SEARCHES

                        ;* THIS IS THE DEFAULT LIST FOR THE FIRST COMMAND LIST ENTRY

F99C                    CMDTBL  EQU     *               ;* MONITOR COMMAND TABLE
F99C  04                        FCB     4
F99D  42                        FCC     'B'             ;* 'BREAKPOINT' COMMAND
F99E  054D                      FDB     CBKPT-*
F9A0  04                        FCB     4
F9A1  43                        FCC     'C'             ;* 'CALL' COMMAND
F9A2  0417                      FDB     CCALL-*
F9A4  04                        FCB     4
F9A5  44                        FCC     'D'             ;* 'DISPLAY' COMMAND
F9A6  049D                      FDB     CDISP-*
F9A8  04                        FCB     4
F9A9  45                        FCC     'E'             ;* 'ENCODE' COMMAND
F9AA  059F                      FDB     CENCDE-*
F9AC  04                        FCB     4
F9AD  47                        FCC     'G'             ;* 'GO' COMMAND
F9AE  03D2                      FDB     CGO-*
F9B0  04                        FCB     4
F9B1  4C                        FCC     'L'             ;* 'LOAD' COMMAND
F9B2  04DD                      FDB     CLOAD-*
F9B4  04                        FCB     4
F9B5  4D                        FCC     'M'             ;* 'MEMORY' COMMAND
F9B6  040D                      FDB     CMEM-*
F9B8  04                        FCB     4
F9B9  4E                        FCC     'N'             ;* 'NULLS' COMMAND
F9BA  04FD                      FDB     CNULLS-*
F9BC  04                        FCB     4
F9BD  4F                        FCC     'O'             ;* 'OFFSET' COMMAND
F9BE  050A                      FDB     COFFS-*
F9C0  04                        FCB     4
F9C1  50                        FCC     'P'             ;* 'PUNCH' COMMAND
F9C2  04AF                      FDB     CPUNCH-*
F9C4  04                        FCB     4
F9C5  52                        FCC     'R'             ;* 'REGISTERS' COMMAND
F9C6  0284                      FDB     CREG-*
F9C8  04                        FCB     4
F9C9  53                        FCC     'S'             ;* 'SETLEVEL' COMMAND
F9CA  04F2                      FDB     CSTLEV-*
F9CC  04                        FCB     4
F9CD  54                        FCC     'T'             ;* 'TRACE' COMMAND
F9CE  04D6                      FDB     CTRACE-*
F9D0  04                        FCB     4
F9D1  56                        FCC     'V'             ;* 'VERIFY' COMMAND
F9D2  04CF                      FDB     CVER-*
F9D4  04                        FCB     4
F9D5  57                        FCC     'W'             ;* 'WINDOW' COMMAND
F9D6  0468                      FDB     CWINDO-*
F9D8  FF                        FCB     -1

                        ;**************************************************************************
                        ;*                      [ SWI FUNCTION 4 AND 5 ]                          *
                        ;*      4       -       OUT2HS  -       DECODE BYTE TO HEX AND ADD SPACE      *
                        ;*      5       -       OUT4HS  -       DECODE WORD TO HEX AND ADD SPACE      *
                        ;* INPUT : X    ->      BYTE OR WORD TO DECODE                            *
                        ;* OUTPUT: CHARACTERS SENT TO OUTPUT HANDLER                              *
                        ;*         X    ->      NEXT BYTE OR WORD                                 *
                        ;**************************************************************************

F9D9                    ZOUT2H
F9D9  A680                      LDA     ,X+             ;* LOAD NEXT BYTE
F9DB  3406                      PSHS    D               ;* SAVE - DO NOT REREAD
F9DD  C610                      LDB     #16             ;* SHIFT BY 4 BITS
F9DF  3D                        MUL                     ;* WITH MULTIPLY
F9E0  8D04                      BSR     ZOUTHX          ;* SEND OUT AS HEX
F9E2  3506                      PULS    D               ;* RESTORE BYTES
F9E4  840F                      ANDA    #$0F            ;* ISOLATE RIGHT HEX
F9E6                    ZOUTHX
F9E6  8B90                      ADDA    #$90            ;* PREPARE A-F ADJUST
F9E8  19                        DAA
F9E9  8940                      ADCA    #$40            ;* PREPARE CHARACTER BITS
F9EB  19                        DAA                     ;* ADJUST
F9EC                    SEND
F9EC  6E9DE5EE                  JMP     [VECTAB+_CODTA,PCR]     ;* SEND TO OUT HANDLER
```

```
F9F0  8DE7            ZOT4HS  BSR     ZOUT2H          ;* CONVERT FIRST BYTE
F9F2  8DE5            ZOT2HS  BSR     ZOUT2H          ;* CONVERT BYTE TO HEX
F9F4  AF64                    STX     4,S             ;* UPDATE USERS X REGISTER

                      ;* FALL INTO SPACE ROUTINE

                      ;*******************************************************************************
                      ;*                               [ SWI FUNCTION 7 ]                            *
                      ;*      SPACE - SEND BLANK TO OUTPUT HANDLER                                   *
                      ;*      INPUT : NONE                                                           *
                      ;*      OUTPUT: BLANK SEND TO CONSOLE HANDLER                                  *
                      ;*******************************************************************************

F9F6                  ZSPACE
F9F6  8620                    LDA     #' '            ;* LOAD BLANK
F9F8  203D                    BRA     ZOTCH2          ;* SEND AND RETURN

                      ;*******************************************************************************
                      ;*                               [ SWI FUNCTION 9 ]                            *
                      ;*      SWAP VECTOR TABLE ENTRY                                                *
                      ;*      INPUT: A=VECTOR CODE (OFFSET)                                          *
                      ;*             X=0 OR REPLACEMENT VALUE                                        *
                      ;*      OUTPUT: X=PREVIOUS VALUE                                               *
                      ;*******************************************************************************

F9FA                  ZVSWTH
F9FA  A661                    LDA     1,S             ;* LOAD REQUESTERS A
F9FC  8134                    CMPA    #HIVTR          ;* ? SUB-CODE TOO HIGH
F9FE  2239                    BHI     ZOTCH3          ;* IGNORE CALL IF SO
FA00  109EC2                  LDY     VECTAB+_AVTBL   ;* LOAD VECTOR TABLE ADDRESS
FA03  EEA6                    LDU     A,Y             ;* U=OLD ENTRY
FA05  EF64                    STU     4,S             ;* RETURN OLD VALUE TO CALLERS X
FA07  AF7E                    STX     -2,S            ;* ? X=0
FA09  272E                    BEQ     ZOTCH3          ;* YES, DO NOT CHANGE ENTRY
FA0B  AFA6                    STX     A,Y             ;* REPLACE ENTRY
FA0D  202A                    BRA     ZOTCH3          ;* RETURN FROM SWI

                      ;*******************************************************************************
                      ;*                               [ SWI FUNCTION 0 ]                            *
                      ;*      INCHNP  -       OBTAIN INPUT CHAR IN A (NO PARITY)                     *
                      ;*      NULLS AND RUBOUTS ARE IGNORED                                          *
                      ;*      AUTOMATIC LINEFEED IS SENT UPON RECEIVING A CARRIAGE RETURN            *
                      ;*      UNLESS WE ARE LOADING FROM TAPE                                        *
                      ;*******************************************************************************

FA0F                  ZINCHP
FA0F  8D5D                    BSR     XQPAUS          ;* REEASE PROCESSOR
FA11                  ZINCH
FA11  8D5F                    BSR     XQCIDT          ;* CALL INPUT DATA APPENDAGE
FA13  24FA                    BCC     ZINCHP          ;* LOOP IF NONE AVAILABLE
FA15  4D                      TSTA                    ;* TEST FOR NULL ?
FA16  27F9                    BEQ     ZINCH           ;* IGNORE NULL
FA18  817F                    CMPA    #$7F            ;* RUBOUT ?
FA1A  27F5                    BEQ     ZINCH           ;* BRANCH YES TO IGNORE
FA1C  A761                    STA     1,S             ;* STORE INTO CALLERS A
FA1E  0D8F                    TST     MISFLG          ;* LOAD IN PROGRESS ?
FA20  2617                    BNE     ZOTCH3          ;* BRANCH IF SO TO NOT EXHO
FA22  810D                    CMPA    #CR             ;* CARRIAGE RETURN ?
FA24  2604                    BNE     ZIN2            ;* NO TEST ECHO BYTE
FA26  860A                    LDA     #LF             ;* LOAD LINEFEED
FA28  8DC2                    BSR     SEND            ;* ALWAYS ECHO LINEFEED
FA2A                  ZIN2
FA2A  0DF4                    TST     VECTAB+_ECHO    ;* ECHO DESIRED ?
FA2C  260B                    BNE     ZOTCH3          ;* NO, RETURN

                      ;*      FALL THRU TO OUTCH

                      ;*******************************************************************************
                      ;*                               [ SWI FUNCTION 1 ]                            *
                      ;*              OUTCH   -       OUTPUT CHARACTER FROM                          *
                      ;*      INPUT : NONE                                                           *
                      ;*      OUTPUT: ID LINEFEED IS THE OUTPUT CHARACTER THEN C=0 NO CTL-X          *
                      ;*              RECEIVED, C=1 CTLX RECEIVED                                    *
                      ;*******************************************************************************

FA2E                  ZOTCH1
FA2E  A661                    LDA     1,S             ;* LOAD CHARACTER TO SEND
FA30  308C09                  LEAX    <ZPCRLS,PCR     ;* DEFAULT FOR LINEFEED
FA33  810A                    CMPA    #LF             ;* LINEFEED ?
FA35  270F                    BEQ     ZPDTLP          ;* BRANCH TO CHECK PAUSE IF SO
FA37                  ZOTCH2
FA37  8DB3                    BSR     SEND            ;* SEND TO OUTPUT ROUTINE
FA39                  ZOTCH3
FA39  0C90                    INC     SWICNT          ;* BUMP UP "SWI" TRACE NEST LEVEL
FA3B  3B                      RTI                     ;* RETURN FROM "SWI" FUNCTIONS

                      ;*******************************************************************************
                      ;*                               [ SWI FUNCTION 6 ]                            *
```

```
                        ;*              PCRLF   -       SEND CR/LF TO CONSOLE HANDLER                *
                        ;*      INPUT : NONE                                                         *
                        ;*      OUTPUT: CR AND LF SENT TO HANDLER                                    *
                        ;*              C=0 NOT CTL-X, C=1 CTL-X RECEIVED                            *
                        ;*****************************************************************************

FA3C                    ZPCRLS
FA3C  04                        FCB     EOT             ;* NULL STRING
FA3D                    ZPCRLF
FA3D  308CFC                    LEAX    ZPCRLS,PCR      ;* READY CR/LF STRING

                        ;*      FALL INTO CR/LF CODE

                        ;*****************************************************************************
                        ;*                              [ SWI FUNCTION 3 ]                           *
                        ;*              PDATA   -       OUTPUT CR/LF AND STRING                      *
                        ;*      INPUT: X -> STRING                                                   *
                        ;*      OUTPUT: CR/LF AND STRING SENT TO OUTPUT CONSOLE HANDLER              *
                        ;*              C=0 NO CTL-X, C=1 CTL-X RECEIVED                             *
                        ;*      NOTE: LINEFEED MUST FOLLOW CARRIAGE RETURN FOR PROPER PUNCH DATA     *
                        ;*****************************************************************************

FA40                    ZPDATA
FA40  860D                      LDA     #CR             ;* LOAD CARRIAGE RETURN
FA42  8DA8                      BSR     SEND            ;* SEND IT
FA44  860A                      LDA     #LF             ;* LOAD LINEFEED

                        ;* FALL INTO PDATA1

                        ;*****************************************************************************
                        ;*                              [ SWI FUNCTION 2 ]                           *
                        ;*      PDATA1 - OUTPUT STRING TILL EOT ($04)                                *
                        ;*      THIS ROUTINE PAUSES ID AN INPUT BYTE BECOMES AVAILABLE DURING OUTPUT  *
                        ;*      TRANSMISION UNTIL A SECODE IS RECEIVED.                              *
                        ;*      INPUT:  X -> STRING                                                  *
                        ;*      OUTPUT: STRING SENT TO OUTPUT CONSOLE DRIVER                         *
                        ;*              C=0 NO CTL-X, C=1 CTL-X RECEIVED                             *
                        ;*****************************************************************************

FA46                    ZPDTLP
FA46  8DA4                      BSR     SEND            ;* SEND CHARACTER TO DRIVER
FA48                    ZPDTA1
FA48  A680                      LDA     ,X+             ;* LOAD NEXT CHARACTER
FA4A  8104                      CMPA    #EOT            ;* EOT ?
FA4C  26F8                      BNE     ZPDTLP          ;* LOOP IF NOT

                        ;* FALL INTO PAUSE CHECK FUNCTION

                        ;*****************************************************************************
                        ;*                              [ WI FUNCTION 12 ]                           *
                        ;*      PAUSE   -       RETURN TO TASK DISPATCHING AND CHECK FOR FREEZE      *
                        ;*                      CONDITION OR CTL-X BREAK.                            *
                        ;*      THIS FUNCTION ENTERS THE TASK PAUSE HANDLER SO OPTIONALLY OTHER       *
                        ;*      6809 PROCESSES MAY GAIN CONTROL. WITH A RESULTING WAIT LOOP, OR      *
                        ;*      CONDITION CODE RETURN ID A CONTROL-X IS ENTERED FROM INPUT HANDLER    *
                        ;*      OUTPUT: C=1 IF CTL-X ENTERED , 0 OTHERWISE                           *
                        ;*****************************************************************************

FA4E                    ZPAUSE
FA4E  8D1E                      BSR     XQPAUS          ;* RELEASE CONTROL AT EVERY LINE
FA50  8D06                      BSR     CHKABT          ;* CHECK FOR FREEZE OR ABORT
FA52  1FA9                      TFR     CC,B            ;* PREPAR FOR FREEZE OR ABPRT
FA54  E7E4                      STB     ,S              ;* OVERLAY OLD ONE ON STACK
FA56  20E1                      BRA     ZOTCH3          ;* RETURN FROM "SWI"

                        ;* CHKABT       -       SCAN FOR INPUT PAUSE/ABORT DURING OUTPUT
                        ;* OUTPUT :     -       C=0 OK, C=1 ABORT ( CTL-X ISSUED )
                        ;* VOLATILE:    -       U,X,D
FA58                    CHKABT
FA58  8D18                      BSR     XQCIDT          ;* ATTEMP INPUT
FA5A  2405                      BCC     CHKRTN          ;* BRANCH NO TO RETURN
FA5C  8118                      CMPA    #CAN            ;* CTL-X FOR ABORT ?
FA5E  2602                      BNE     CHKWT           ;* BRANCH NOT TO PAUSE
FA60                    CHKSEC
FA60  53                        COMB                    ;* SET CARRY
FA61                    CHKRTN
FA61  39                        RTS                     ;* RETURN TO CALLER WITH CC SET
FA62                    CHKWT
FA62  8D0A                      BSR     XQPAUS          ;* PAUSE FOR A MOMENT
FA64  8D0C                      BSR     XQCIDT          ;* KEY FOR START ?
FA66  24FA                      BCC     CHKWT           ;* LOOP UNTIL RECEIVED
FA68  8118                      CMPA    #CAN            ;* ABORT SIGNALLED FROM WAIT
FA6A  27F4                      BEQ     CHKSEC          ;* BRANCH YES
FA6C  4F                        CLRA                    ;* SET C=0 FOR NO ABORT
FA6D  39                        RTS

                        ;*      SAVE MEMORY WITH JUMPS
```

```
FA6E  6E9DE578        XQPAUS  JMP     [VECTAB+_PAUSE,PCR]     ;* TO PAUSE ROUTINE
FA72  AD9DE562        XQCIDT  JSR     [VECTAB+_CIDTA,PCR]     ;* TO INPUT ROUTINE
FA76  847F                    ANDA    #$7F                    ;* STRIP PARITY
FA78  39                      RTS                             ;* AND RETURN

                      ;*******************************************************************************
                      ;*                       NMI DEFAULT INTERRUPT HANDLER                         *
                      ;*      THE NMI HANDLER IS USED FOR TRACING INSTRUCTIONS. TRACE PRINTOUTS      *
                      ;*      OCCURS ONLY AS LONG AS THE STACK TRACE LEVEL IS NOT BREACHED BY        *
                      ;*      FALLING BELOW IT. TRACING CONTINUES UNTIL THE COUNT TURNS ZERO OR      *
                      ;*      A CTL-X IS ENTERED FROM THE INPUT CONSOLE DEVICE.                      *
                      ;*******************************************************************************

FA79  4F502D04        MSHOWP  FCB     'O','P','-',EOT ;* OPCODE PREP
FA7D                  NMIR
FA7D  8D42                    BSR     LDDP            ;* LOAD PAGE AND VERIFY STACK
FA7F  0D8F                    TST     MISFLG          ;* THRU A BREAKPOINT ?
FA81  2634                    BNE     NMICON          ;* BRANCH IF SO TO CONTINUE
FA83  0D90                    TST     SWICNT          ;* INHIBIT "SWI" DURING TRACE ?
FA85  2B29                    BMI     NMITRC          ;* BRANCH YES
FA87  306C                    LEAX    12,S            ;* OBTAIN USER STACK POINTERS
FA89  9CF8                    CMPX    SLEVEL          ;* TO TRACE HERE
FA8B  2523                    BLO     NMITRC          ;* BRANCH IF TOO LOW TO DISPLAY
FA8D  308CE9                  LEAX    MSHOWP,PCR      ;* LOAD OP PREP
FA90  3F                      SWI                     ;* SEND TO CONSOLE
FA91  02                      FCB     PDATA1          ;* FUNCTION
FA92  098E                    ROL     DELIM           ;* SAVE CARRY BIT
FA94  308DE501                LEAX    LASTOP,PCR      ;* POINT TO LAST OP
FA98  3F                      SWI                     ;* SEND OUT AS HEX
FA99  05                      FCB     OUT4HS          ;* FUNCTION
FA9A  8D17                    BSR     REGPRS          ;* FOLLOW MEMORY WITH REGISTERS
FA9C  2537                    BCS     ZBKCMD          ;* BRANCH IF "CANCEL"
FA9E  068E                    ROR     DELIM           ;* RESTORE CARRY BIT
FAA0  2533                    BCS     ZBKCMD          ;* BRANCH IF "CANCEL"
FAA2  9E91                    LDX     TRACEC          ;* LOAD TRACE COUNT
FAA4  272F                    BEQ     ZBKCMD          ;* IF ZERO TO COMMAND HANDLER
FAA6  301F                    LEAX    -1,X            ;* MINUS ONE
FAA8  9F91                    STX     TRACEC          ;* REFRESH
FAAA  2729                    BEQ     ZBKCMD          ;* STOP TRACE WHEN ZERO
FAAC  8DAA                    BSR     CHKABT          ;* ABORT THE TRACE ?
FAAE  2525                    BCS     ZBKCMD          ;* BRANCH YES TO COMMAND HANDLER
FAB0                  NMITRC
FAB0  1603F7                  LBRA    CTRCE3          ;* NOMTRACE ANOTHER INSTRUCTION
FAB3                  REGPRS
FAB3  1701B9                  LBSR    REGPRT          ;* PRINT REGISTERS AS FROM COMMAND
FAB6  39                      RTS                     ;* RETURN TO CALLER

                      ;* JUST EXECUTED THRU A BREAKPOINT
                      ;* NOW CONTINUE NORMALLY

FAB7                  NMICON
FAB7  0F8F                    CLR     MISFLG          ;* CLEAR THRU FLAG
FAB9  1702EB                  LBSR    ARMBK2          ;* ARM BREAKPOINTS
FABC                  RTI
FABC  3B                      RTI                     ;* AND CONTINUE USERS PROGRAM

                      ;*      LDDP    -       SETUP DIRECT PAGE REGISTERS, VERIFY STACK
                      ;*      AN INVALID STACK CAUSES A RETURN TO THE COMMAND HANDLER
                      ;*      INPUT: FULLY STACKED REGISTERS FROM AN INTERRUPT
                      ;*      OUTPUT: DPR LOADED TO WORK PAGE

FABD  3F072004        ERRMSG  FCB     "?",BELL,$20,EOT        ;* ERROR RESPONSE

FAC1                  LDDP
FAC1  E68DE4D8                LDB     BASEPG,PCR      ;* LOAD DIRECT PAGE HIGH BYTE
FAC5  1F9B                    TFR     B,DP            ;* SETUP DIRECT PAGE REGISTER
FAC7  A163                    CMPA    3,S             ;* STACK VALID
FAC9  2725                    BEQ     RTS             ;* YES RETURN
FACB  10DE97                  LDS     RSTACK          ;* RESET TO INITIAL STACK POINTER
FACE                  ERROR
FACE  308CEC                  LEAX    ERRMSG,PCR      ;* LOAD ERROR REPORT
FAD1  3F                      SWI                     ;* SEND OUT ERROR REPORT
FAD2  03                      FCB     PDATA           ;* ON NEXT LINE

                      ;* FALL INTO BREAKPOINT HANDLER

                      ;*******************************************************************************
                      ;*                              [ SWI FUNCTION 10 ]                            *
                      ;*                      BREAKPOINT PROGRAM FUNCTION                            *
                      ;*      PRINT REGISTERS AND GOTO COMMAND HANDLER                               *
                      ;*******************************************************************************

FAD3  8DDE            ZBKPNT  BSR     REGPRS          ;* PRINT OUT REGISTERS
FAD5  16FE21          ZBKCMD  LBRA    CMDNEP          ;* NOW ENTER COMMAND HANDLER

                      ;*******************************************************************************
                      ;*              IRQ RESERVED, SWI2 AND SWI3 INTERRUPT HANDLERS                 *
                      ;*      THE DEFAULT HANDLING IS TO CAUSE A BREAKPOINT                          *
```

```
                        ;*******************************************************************************

FAD8                    SWI2R   EQU     *               ;* SWI2 ENTRY
FAD8                    SWI3R   EQU     *               ;* SWI3 ENTRY
FAD8                    IRQR    EQU     *               ;* IRQ ENTRY
FAD8  8DE7              RSRVDR  BSR     LDDP            ;* SET BASE PAGE , VALIDATE STACK
FADA  20F7                      BRA     ZBKPNT          ;* FORCE A BREAKPOINT

                        ;*******************************************************************************
                        ;*                       FIRQ HANDLER                                          *
                        ;*      JUST RETURN FOR FIRQ INTERRUPT                                         *
                        ;*******************************************************************************

FABC                    FIRQR   EQU     RTI             ;* IMMEDIATE RETURN

                        ;*******************************************************************************
                        ;*              DEFAULT I/O DRIVERS                                            *
                        ;*******************************************************************************

                        ;*      CIDTA   -       RETURN CONSOLE INPUT CHARACTER
                        ;*      OUTPUT:  C=0 IF NO DATA READY, C=1 A=CHARACTER
                        ;*      U VOLATILE

FADC                    CIDTA
FADC  DEF0                      LDU     VECTAB+_ACIA    ;* LOAD ACIA ADDRESS
FADE  A6C4                      LDA     ,U              ;* LOAD STATUS REGISTER
FAE0  44                        LSRA                    ;* TEST RECEIVER REGISTER FLAG
FAE1  2402                      BCC     CIRTN           ;* RETURN IF NOTHING
FAE3  A641                      LDA     1,U             ;* LOAD DATA BYTE
FAE5                    CIRTN
FAE5  39                        RTS                     ;* RETURN TO CALLER

                        ;* CION -       INPUT CONSOLE INITALIZATION
                        ;* COON -       OUTPUT CONSOLE INITALIZATION
                        ;* A,X  VOLATILE

FAE6                    CION    EQU     *
FAE6                    COON
FAE6  8603                      LDA     #3              ;* ACIA RESET CODE
FAE8  9EF0                      LDX     VECTAB+_ACIA    ;* LOAD ACIA ADDRESS
FAEA  A784                      STA     0,X             ;* STORE INTO STATUS REGISTER
FAEC  8651                      LDA     #$51            ;* SET CONTROL
                        ;*      LDA     #$11            ;* WON'T HOLD UP HANDSHAKING
FAEE  A784                      STA     ,X              ;* REGISTER UP
FAF0                    RTS
FAF0  39                        RTS                     ;* RETURN TO CALLER

                        ;* THE FOLLOWING HAVE NO DUTIES TO PERFORM

FAF0                    CIOFF   EQU     RTS             ;* CONSOLE INPUT OFF
FAF0                    COOFF   EQU     RTS             ;* CONSOLE OUTPUT OFF

                        ;* CODTA        -       OUTPUT CHARACTER TO CONSOLE DEVICE
                        ;* INPUT : A = CHARACTER TO SEND
                        ;* OUTPUT: CHAR SENT TO TERMINAL WITH PROPER PADDING
                        ;* ALL REGISTEREX TRANSPARENT

FAF1                    CODTA
FAF1  3447                      PSHS    U,D,CC          ;* SAVE REGISTERS, WORK BYTE
FAF3  DEF0                      LDU     VECTAB+_ACIA    ;* LOAD ACIA ADDRESS
FAF5  8D1B                      BSR     CODTAO          ;* CALL OUTPUT CHAR SUBROUTINE
FAF7  8110                      CMPA    #DLE            ;* DATA LINE ESCAPE
FAF9  2712                      BEQ     CODTRT          ;* YES, RETURN
FAFB  D6F2                      LDB     VECTAB+_PAD     ;* DELAULT TO CHAR PAD COUNT
FAFD  810D                      CMPA    #CR             ;* CR ?
FAFF  2602                      BNE     CODTPD          ;* BRANCH SO
FB01  D6F3                      LDB     VECTAB+_PAD+1   ;* LOAD NEW LINE PAD COUNT
FB03                    CODTPD
FB03  4F                        CLRA                    ;* CREATE NULL
FB04  E7E4                      STB     ,S              ;* SAVE COUNT
FB06  8C                        FCB     SKIP2           ;* ENTERLOOP
FB07                    CODTLP
FB07  8D09                      BSR     CODTAO          ;* SEND NULL
FB09  6AE4                      DEC     ,S              ;* FINISHED ?
FB0B  2AFA                      BPL     CODTLP          ;* NO, CONTINUE WITH MORE
FB0D                    CODTRT
FB0D  35C7                      PULS    PC,U,D,CC       ;* RESTORE REGISTERS AND RETURN
FB0F                    CODTAD
FB0F  17FF5C                    LBSR    XQPAUS          ;* TEMPORARY GIVE UP CONTROL
FB12                    CODTAO
FB12  E6C4                      LDB     ,U              ;* LOAD ACIA CONTROL REGISTERS
FB14  C502                      BITB    #$02            ;* TX REGISTER EMPTY ?
FB16  27F7                      BEQ     CODTAD          ;* RELEASE CONTROL IF NOT
FB18  A741                      STA     1,U             ;* STORE INTO DATA REGISTER
FB1A  39                        RTS                     ;* RETURN TO CALLER

                        ;* E
```

```
                        ;*      BSON    -       TURN ON READ/VERIFY/PUNCH MECHANISM
                        ;*      A IS VOLATILE

FB1B                    BSON
FB1B  8611                      LDA     #$11            ;* SET READ CODE
FB1D  6D66                      TST     6,S             ;* READ OR VERIFY ?
FB1F  2601                      BNE     BSON2           ;* BRANCH YES
FB21  4C                        INCA
FB22                    BSON2
FB22  3F                        SWI                     ;* PERFORM OUTPUT
FB23  01                        FCB     OUTCH           ;* FUNCTION
FB24  0C8F                      INC     MISFLG          ;* SET LOAD IN PROGRESS FLAG
FB26  39                        RTS                     ;* RETURN TO CALLER

                        ;* BSOFF - TURN OFF READ/VERIFY/PUNCH MECHANISM
                        ;* A,X - VOLATILE

FB27                    BSOFF
FB27  8614                      LDA     #$14            ;* TO DC4 - STOP
FB29  3F                        SWI                     ;* SEND OUT
FB2A  01                        FCB     OUTCH           ;* FUNCTION
FB2B  4A                        DECA                    ;* CHANGE TO DC3 (X-OFF)
FB2C  3F                        SWI                     ;* SEND OUT
FB2D  01                        FCB     OUTCH           ;* FUNCTION
FB2E  0A8F                      DEC     MISFLG          ;* CLEAR LOAD IN PROGRESS FLAG
FB30  8E61A8                    LDX     #25000          ;* DELAY 1 SECOND (2 MHz clock)
FB33                    BSOFLP
FB33  301F                      LEAX    -1,X            ;* COUNT DOWN
FB35  26FC                      BNE     BSOFLP          ;* LOOP UNTIL DONE
FB37  39                        RTS                     ;* RETURN TO CALLER

                        ;* BSDTA - READ/VERIFY/PUNCH HANDLER
                        ;* INPUT: S+6 = CODE BYTE , VERIFY (-1) , PUNCH (0) , LOAD (1)
                        ;*        S+4 = START ADDRESS
                        ;*        S+2 = STOP ADDRESS
                        ;*        S+0 = RETURN ADDRESS
                        ;* OUTPUT: Z=1 NORMAL COMPLETION, Z=0 INVALID LOAD/VER
                        ;* REGISTERS ARE VOLATILE

FB38                    BSDTA
FB38  EE62                      LDU     2,S             ;* U = TO ADDRESS OR OFFSET
FB3A  6D66                      TST     6,S             ;* PUNCH ?
FB3C  2754                      BEQ     BSDPUN          ;* BRANCH YES

                        ;* DURING READ/VERIFY:  S+2 = MSB ADDRESS SAVE BYTE
                        ;*                      S+1 = BYTE COUNTER
                        ;*                      S+0 = CHECKSUM
                        ;*                      U HOLDS OFFSET

FB3E  327D                      LEAS    -3,S            ;* ROOM FOR WORK/COUNTER/CHECKSUM
FB40                    BSDLD1
FB40  3F                        SWI                     ;* GET NEXT CHARACTER
FB41  00                        FCB     INCHNP          ;* FUNCTION
FB42                    BSDLD2
FB42  8153                      CMPA    #'S'            ;* START OF S1/S9
FB44  26FA                      BNE     BSDLD1          ;* BRANCH NOT
FB46  3F                        SWI                     ;* GET NEXT CHARACTER
FB47  00                        FCB     INCHNP          ;* FUNCTION
FB48  8139                      CMPA    #'9'            ;* HAVE S9 ?
FB4A  2722                      BEQ     BSDSRT          ;* YES , RETURN GOOD CODE
FB4C  8131                      CMPA    #'1'            ;* HAVE NEW RECORD
FB4E  26F2                      BNE     BSDLD2          ;* BRANCH IF NOT
FB50  6FE4                      CLR     ,S              ;* CLEAR CHECKSUM
FB52  8D21                      BSR     BYTE            ;* OBTAIN BYTE COUNT
FB54  E761                      STB     1,S             ;* SAVE FOR DECREMENT
                        ;* READ ADDRESS
FB56  8D1D                      BSR     BYTE            ;* OBTAIN HIGH VALUE
FB58  E762                      STB     2,S             ;* SAVE IT
FB5A  8D19                      BSR     BYTE            ;* OBTAIN LOW VALUE
FB5C  A662                      LDA     2,S             ;* MAKE D=VALUE
FB5E  31CB                      LEAY    D,U             ;* Y= ADDRESS + OFFSET
                        ;* STORE TEXT
FB60                    BSDNXT
FB60  8D13                      BSR     BYTE            ;* NEXT BYTE
FB62  270C                      BEQ     BSDEOL          ;* BRANCH IF CHECKSUM
FB64  6D69                      TST     9,S             ;* VERIFY ERROR ?
FB66  2B02                      BMI     BSDCMP          ;* YES, ONLY COMPARE
FB68  E7A4                      STB     ,Y              ;* STORE INTO MEMORY
FB6A                    BSDCMP
FB6A  E1A0                      CMPB    ,Y+             ;* VALID RAM
FB6C  27F2                      BEQ     BSDNXT          ;* YES, CONTINUE READING
FB6E                    BSDSRT
FB6E  3592                      PULS    PC,X,A          ;* RETURN WITH A Z SET PROPER

FB70                    BSDEOL
FB70  4C                        INCA                    ;* CALID CHECKSUM ?
FB71  27CD                      BEQ     BSDLD1          ;* BRANCH YES
FB73  20F9                      BRA     BSDSRT          ;* RETURN Z=0 INVALID
```

```
                    ;* BYTE BULIDS 8 NIT VALUE FROM TWO HEX DIGITS IN
FB75                BYTE
FB75  8D12                  BSR     BYTHEX          ;* OBTAI FIRST HEX
FB77  C610                  LDB     #16             ;* PREPARE SHIFT
FB79  3D                    MUL                     ;* OVER TO A
FB7A  8D0D                  BSR     BYTHEX          ;* OBTAIN SECOND HEX
FB7C  3404                  PSHS    B               ;* SAVE HIGH HEX
FB7E  ABE0                  ADDA    ,S+             ;* COMBINE BOTH SIDES
FB80  1F89                  TFR     A,B             ;* SEND BACK IN B
FB82  AB62                  ADDA    2,S             ;* COMPUTE NEW CHECKSUM
FB84  A762                  STA     2,S             ;* STORE BACK
FB86  6A63                  DEC     3,S             ;* DECREMENT BYTE COUNT
FB88                BYTRTS
FB88  39                    RTS                     ;* RETURN TO CALLER
FB89                BYTHEX
FB89  3F                    SWI                     ;* GET NEXT HEX
FB8A  00                    FCB     INCHNP          ;* CHARACTER
FB8B  1701D4                LBSR    CNVHEX          ;* CONVERT TO HEX
FB8E  27F8                  BEQ     BYTRTS          ;* RETURN IF VALID HEX
FB90  35F2                  PULS    PC,U,Y,X,A      ;* RETURN TO CALLER WITH Z=0

                    ;* PUNCH STACK USE :    S+8 = TO ADDRESS
                    ;*                      S+6 = RETURN ADDRESS
                    ;*                      S+4 = SAVED PADDING VALUES
                    ;*                      S+2 = FROM ADDRES
                    ;*                      S+1 = FRAME COUNT/CHECKSUM
                    ;*                      S+0 = BYTE COUNT

FB92                BSDPUN
FB92  DEF2                  LDU     VECTAB+_PAD     ;* LOAD PADDINT VALUES
FB94  AE64                  LDX     4,S             ;* X= FROM ADDRESS
FB96  3456                  PSHS    U,X,D           ;* CREATE STACK WORK AREA
FB98  CC0018                LDD     #24             ;* SET A=0 , B=24
                    ;* PAG 018
FB9B  D7F2                  STB     VECTAB+_PAD     ;* SETUP 24 CHARACTER PADS
FB9D  3F                    SWI                     ;* SEND NULLS OUT
FB9E  01                    FCB     OUTCH           ;* FUNCTION
FB9F  C604                  LDB     #4              ;* SETUP NEW LINE PAD TO 4
FBA1  DDF2                  STD     VECTAB+_PAD     ;* SET UP PUNCH PADDING
                    ;* CALCULATE SIZE
FBA3                BSPGO
FBA3  EC68                  LDD     8,S             ;* LOAD TO
FBA5  A362                  SUBD    2,S             ;* MINUS FROM=LEHGTH
FBA7  10830018              CMPD    #24             ;* MORE THAN 23 ?
FBAB  2502                  BLO     BSPOK           ;* NO, OK
FBAD  C617                  LDB     #23             ;* FORCE TO 23 MAX
FBAF                BSPOK
FBAF  5C                    INCB                    ;* PREPARE COUNETER
FBB0  E7E4                  STB     ,S              ;* STORE BYTE COUNT
FBB2  CB03                  ADDB    #3              ;* ADJUST TO FRAME COUNT
FBB4  E761                  STB     1,S             ;* SAVE
                    ;* PUNCH CR,LF,NULS,S,1
FBB6  308C33                LEAX    <BSPSTR,PCR     ;* LOAD START RECORD HEADER
FBB9  3F                    SWI                     ;* SEND OUT
FBBA  03                    FCB     PDATA           ;* FUNCTION
                    ;* SEND FRAME COUNT
FBBB  5F                    CLRB                    ;* INITIALIZE CHECKSUB
FBBC  3061                  LEAX    1,S             ;* POINT TO FRAME COUNT AND ADDR
FBBE  8D27                  BSR     BSPUN2          ;* SEND FRAME COUNT
                    ;* DATA ADDRESS
FBC0  8D25                  BSR     BSPUN2          ;* SEND ADDRESS HI
FBC2  8D23                  BSR     BSPUN2          ;* SEND ADDRESS LOW
                    ;* PUNCH DATA
FBC4  AE62                  LDX     2,S             ;* LOAD START DATA ADDRESS
FBC6                BSPMRE
FBC6  8D1F                  BSR     BSPUN2          ;* SEND OUT NEXT BYTE
FBC8  6AE4                  DEC     ,S              ;* FINAL BYTE ?
FBCA  26FA                  BNE     BSPMRE          ;* LOOP IF NOT DONE
FBCC  AF62                  STX     2,S             ;* UPDATE FROM ADDRESS VALUE
                    ;* PUNCH CHECKSUM
FBCE  53                    COMB                    ;* COMPLEMENT
FBCF  E761                  STB     1,S             ;* STORE FOR SENDOUT
FBD1  3061                  LEAX    1,S             ;* POINT TO IT
FBD3  8D14                  BSR     BSPUNC          ;* SEND OUT AS HEX
FBD5  AE68                  LDX     8,S             ;* LOAD TOP ADDRESS
FBD7  AC62                  CMPX    2,S             ;* DONE ?
FBD9  24C8                  BHS     BSPGO           ;* BRANCH NOT
FBDB  308C11                LEAX    <BSPEOF,PCR     ;* PREPARE END OF FILE
FBDE  3F                    SWI                     ;* SEND OUT STRING
FBDF  03                    FCB     PDATA           ;* FUNCTION
FBE0  EC64                  LDD     4,S             ;* RECOVER PAD COUNTS
FBE2  DDF2                  STD     VECTAB+_PAD     ;* RESTORE
FBE4  4F                    CLRA                    ;* SET Z=1 FOR OK RETURN
FBE5  35D6                  PULS    PC,U,X,D        ;* RETURN WITH OK CODE
FBE7                BSPUN2
FBE7  EB84                  ADDB    ,X              ;* ADD TO CHECKSUM
FBE9                BSPUNC
FBE9  16FDED                LBRA    ZOUT2H          ;* SEND OUT AS HEX AND RETURN
```

```
FBEC                    BSPSTR
FBEC  533104                    FCB     'S','1',EOT     ;* CR,LF,NULLS,S,1
FBEF                    BSPEOF
FBEF  53393033303030304643          FCC     /S9030000FC/    ;* EOF STRING
FBF9  0D0A04                    FCB     CR,LF,EOT

                        ;* HSDTA - HIGH SPEED PRINT MEMORY

                        ;* INPUT :       S+4     =       START ADDRESS
                        ;* OUTPUT:       S+2     =       STOP ADDRESS
                        ;*               S+0     =       RETURN ADDRESS
                        ;* X,D VOLATILE

                        ;* SEND TITLE

FBFC                    HSDTA
FBFC  3F                        SWI                     ;* SEND NEW LINE FUNCTION
FBFD  06                        FCB     PCRLF           ;* FUNCTION
FBFE  C606                      LDB     #6              ;* PREPARE 6 SPACES
FC00                    HSBLNK
FC00  3F                        SWI                     ;* SEND BLANK
FC01  07                        FCB     SPACE           ;* FUNCTION
FC02  5A                        DECB                    ;* COUNT DOWN
FC03  26FB                      BNE     HSBLNK          ;* LOOP IF MORE
FC05  5F                        CLRB                    ;* SETUP BYTE COUNT
FC06                    HSHTTL
FC06  1F98                      TFR     B,A             ;* PREPARE FOR CONVERT
FC08  17FDDB                    LBSR    ZOUTHX          ;* CONVERT TO A HEX DIGIT
FC0B  3F                        SWI                     ;* SEND BLANK
FC0C  07                        FCB     SPACE           ;* FUNCTION
FC0D  3F                        SWI                     ;* SEND ANOTHER
FC0E  07                        FCB     SPACE           ;* BLANK
FC0F  5C                        INCB                    ;* UP ANOTHER
FC10  C110                      CMPB    #$10            ;* PASE 'F' ?
FC12  25F2                      BLO     HSHTTL          ;* LOOP UNTIL SO
FC14                    HSHLNE
FC14  3F                        SWI                     ;* TO NEXT LINE
FC15  06                        FCB     PCRLF           ;* FUNCTION
FC16  252F                      BCS     HSDRTN          ;* RETURN IF USER ENTERED CTL-X
FC18  3064                      LEAX    4,S             ;* POINT AT ADDRESS TO CONVERT
FC1A  3F                        SWI                     ;* PRINT OUT ADDRESS
FC1B  05                        FCB     OUT4HS          ;* FUNCTION
FC1C  AE64                      LDX     4,S             ;* LOAD ADDRESS PROPER
FC1E  C610                      LDB     #16             ;* NEXT SIXTEEN
FC20                    HSHNXT
FC20  3F                        SWI                     ;* CONVERT BYTE TO HEX AND SEND
FC21  04                        FCB     OUT2HS          ;* FUNCTION
FC22  5A                        DECB                    ;* COUNTDOWN
FC23  26FB                      BNE     HSHNXT          ;* LOOP IF NOT SIXTEENTH
FC25  3F                        SWI                     ;* SEND BLANK
FC26  07                        FCB     SPACE           ;* FUNCTION
FC27  AE64                      LDX     4,S             ;* RELOAD FROM ADDRESS
FC29  C610                      LDB     #16             ;* COUNT
FC2B                    HSHCHR
FC2B  A680                      LDA     ,X+             ;* NEXT BYTE
FC2D  2B04                      BMI     HSHDOT          ;* TOO LARGE, TO A DOT
FC2F  8120                      CMPA    #' '            ;* LOWER THAN A BLANK ?
FC31  2402                      BHS     HSHCOK          ;* NO BRANCH OK
FC33                    HSHDOT
FC33  862E                      LDA     #'.'            ;* CONVERT INVALID TO A DOT
FC35                    HSHCOK
FC35  3F                        SWI                     ;* SEND CHARACTER
FC36  01                        FCB     OUTCH           ;* FUNCTION
FC37  5A                        DECB                    ;* DONE ?
FC38  26F1                      BNE     HSHCHR          ;* BRANCH NO
FC3A  AC62                      CMPX    2,S             ;* PAST LAST ADDRESS
FC3C  2409                      BHS     HSDRTN          ;* QUIT IF SO
FC3E  AF64                      STX     4,S             ;* UPDATE FROM ADDRESS
FC40  A665                      LDA     5,S             ;* LOAD LOW BYTE ADDRESS
FC42  48                        ASLA                    ;* TO SECTION BOUNDARY ?
FC43  26CF                      BNE     HSHLNE          ;* BRANCH IF NOT
FC45  20B5                      BRA     HSDTA           ;* BRABCH IF SO
FC47                    HSDRTN
FC47  3F                        SWI                     ;* SEND NEW LINE
FC48  06                        FCB     PCRLF           ;* FUNCTION
FC49  39                        RTS                     ;* RETURN TO CALLER

                        ;* F

                        ;*******************************************************************************
                        ;*                       A S S I S T 0 9   C O M M A N D S                     *
                        ;*******************************************************************************

                        ********************* REGISTERS - DISPLAY AND CHANGE REGISTERS

FC4A                    CREG
FC4A  8D23                      BSR     REGPRT          ;* PRINT REGISTERS
FC4C  4C                        INCA                    ;* SEE FOR CHANGE FUNCTION
```

```
FC4D  8D21                BSR     REGCHG          ;* NOCHANGE, DISPLAY REGISTERS
FC4F  39                  RTS                     ;* RETURN TO COMMAND PROCESSOR

                  ;*********************************************************************************
                  ;*              REGPRT  -       PRINT/CHANGE REGISTERS SUBROUTINE                *
                  ;*      WILL ABORT TO 'COMBAD' IF OVERFLOW DETECTED DURING A CHANGE               *
                  ;*      OPERATION, CHANGE DISPLAYS REGISTERS WHEN DONE.                          *
                  ;*                                                                               *
                  ;*      REGISTER MASK LIST CONSISTS OF:                                          *
                  ;*      A> CHARACTERS DENOTING REGISTER                                          *
                  ;*      B> ZERO FOR ONE BYTE, -1 FOR TWO                                         *
                  ;*      C> OFFSET ON STACK TO REGISTER POSITION                                  *
                  ;*      INPUT : SP+4 = STACKED REGISTERS                                         *
                  ;*              A = 0 PRINT, A # 0 PRINT AND CHANGE                              *
                  ;*      OUTPUT: (ONLY FOR REGISTER DISPLAY)                                      *
                  ;*              C=1 CONTROL-X ENTERED, C=0 OTHERWISE                             *
                  ;*      VOLATILE: D,X (CHANGE)                                                   *
                  ;*                B,X (DISPLAY)                                                  *
                  ;*********************************************************************************

FC50              REGMSK
FC50  5043FF13            FCB     'P','C',-1,19   ;* PROGRAM COUNTER
FC54  41000A              FCB     'A',0,10        ;* ACCA
FC57  42000B              FCB     'B',0,11        ;* ACCB
FC5A  58FF0D              FCB     'X',-1,13       ;* INDEX X
FC5D  59FF0F              FCB     'Y',-1,15       ;* INDEX Y
FC60  55FF11              FCB     'U',-1,17       ;* U STACK
FC63  53FF01              FCB     'S',-1,1        ;* S STACK
FC66  43430009            FCB     'C','C',0,9     ;* CC REG
FC6A  4450000C            FCB     'D','P',0,12    ;* DP REG
FC6E  00                  FCB     0               ;* END OF LIST

FC6F              REGPRT
FC6F  4F                  CLRA                    ;* SETUP PRINT ONLY FLAG
FC70              REGCHG
FC70  30E810              LEAX    4+12,S          ;* READY STACK VALUE
FC73  3432                PSHS    Y,X,A           ;* SAVE ON STACK WITH OPTION
FC75  318CD8              LEAY    REGMSK,PCR      ;* LOAD REGISTER MASK
FC78              REGP1
FC78  ECA0                LDD     ,Y+             ;* LOAD NEXT CHARACTER OF <=0
FC7A  4D                  TSTA                    ;* END OF CHARACTERS ?
FC7B  2F04                BLE     REGP2           ;* BRANCH NOT CHARACTER
FC7D  3F                  SWI                     ;* SEND TO CONSOLE
FC7E  01                  FCB     OUTCH           ;* FUNCTION BYTE
FC7F  20F7                BRA     REGP1           ;* CHECK NEXT
FC81              REGP2
FC81  862D                LDA     #'-'            ;* READY '-'
FC83  3F                  SWI                     ;* SEND OUT
FC84  01                  FCB     OUTCH           ;* WITH OUTCH
FC85  30E5                LEAX    B,S             ;* X -> REGISRER TO PRINT
FC87  6DE4                TST     ,S              ;* CHANGE OPTION ?
FC89  2612                BNE     REGCNG          ;* BRANCH YES
FC8B  6D3F                TST     -1,Y            ;* ONE OR TWO BYTES
FC8D  2703                BEQ     REGP3           ;* BRANCH ZERO MEANS ONE
FC8F  3F                  SWI                     ;* PERFORM WORD HEX
FC90  05                  FCB     OUT4HS          ;* FUNCTION
FC91  8C                  FCB     SKIP2           ;* SKIP BYTE PRINT
FC92              REGP3
FC92  3F                  SWI                     ;* PERFORM BYTE HEX
FC93  04                  FCB     OUT2HS          ;* FUNCTION
FC94              REG4
FC94  ECA0                LDD     ,Y+             ;* TO FRONT OF NEXT ENTRY
FC96  5D                  TSTB                    ;* END OF ENTRIES ?
FC97  26DF                BNE     REGP1           ;* LOOP ID MORE
FC99  3F                  SWI                     ;* FORCE NEW LINE FUNCTION
FC9A  06                  FCB     PCRLF           ;* FUNCTION
FC9B              REGRTN
FC9B  35B2                PULS    PC,Y,X,A        ;* RESTORE STACK AND RETURN
FC9D              REGCNG
FC9D  8D40                BSR     BLDNNB          ;* INPUT BINARY NUMBER
FC9F  2710                BEQ     REGNXC          ;* IF CHANGE THEN JUMP
FCA1  810D                CMPA    #CR             ;* NO MORE DESIRED ?
FCA3  271E                BEQ     REGAGN          ;* BRANCH NOPE
FCA5  E63F                LDB     -1,Y            ;* LOAD SIZE FLAG
FCA7  5A                  DECB                    ;* MINUS ONE
FCA8  50                  NEGB                    ;* MAKE POSITIVE
FCA9  58                  ASLB                    ;* TIMES TWO (=2 OR =4)
FCAA              REGSKP
FCAA  3F                  SWI                     ;* PERFORM SPACES
FCAB  07                  FCB     SPACE           ;* FUNCTION
FCAC  5A                  DECB
FCAD  26FB                BNE     REGSKP          ;* LOOP IF MORE
FCAF  20E3                BRA     REG4            ;* CONTINUE WITH NEXT REGISTER
FCB1              REGNXC
FCB1  A7E4                STA     ,S              ;* SAVE DELIMITER IN OPTION (ALWAYS >0)
FCB3  DC9B                LDD     NUMBER          ;* OBTAIN BINARY RESULT
FCB5  6D3F                TST     -1,Y            ;* TWO BYTES WORTH ?
FCB7  2602                BNE     REGTWO          ;* BRANCH YES
```

```
FCB9  A682                LDA     ,-X             ;* SETUP FOR TWO
FCBB                  REGTWO
FCBB  ED84                STD     ,X              ;* STORE IN NEW LINE
FCBD  A6E4                LDA     ,S              ;* RECOVER DELIMITER
FCBF  810D                CMPA    #CR             ;* END OF CHANGE ?
FCC1  26D1                BNE     REG4            ;* NO, KEEP ON TRUCK'N
                      ;* MOVE STACKED DATA TO NEW STACK IN CASE STACK
                      ;* POINTER HAS CHANGED

FCC3                  REGAGN
FCC3  308DE28A            LEAX    TSTACK,PCR      ;* LOAD TEMP AREA
FCC7  C615                LDB     #21             ;* LOAD COUNT
FCC9                  REGTF1
FCC9  3502                PULS    A               ;* NEXT BYTE
FCCB  A780                STA     ,X+             ;* STORE INTO TEMP
FCCD  5A                  DECB                    ;* COUNT DOWN
FCCE  26F9                BNE     REGTF1          ;* LOOP IF MORE
FCD0  10EE88EC            LDS     -20,X           ;* LOAD NEW STACK POINTER
FCD4  C615                LDB     #21             ;* LOAD COUNT AGAIN
FCD6                  REGTF2
FCD6  A682                LDA     ,-X             ;* NEXT TO STORE
FCD8  3402                PSHS    A               ;* BACK ONTO NEW STACK
FCDA  5A                  DECB                    ;* COUNT DOWN
FCDB  26F9                BNE     REGTF2          ;* LOOP IF MORE
FCDD  20BC                BRA     REGRTN          ;* GO RESTART COMMAND

                      ;*************************************************************************
                      ;*              BLDNUM - BUILDS BINARY VALUE FROM INPUT HEX              *
                      ;*      THE ACTIVE EXPRESSION HANDLER IS USED.                           *
                      ;*      INPUT : S = RETURN ADDRESS                                       *
                      ;*      OUTPUT: A = DELIMITER WHICH TERMINATED VALUE (IF DELM NOT ZERO)  *
                      ;*      "NUMBER" = WORD BINARY RESULT                                    *
                      ;*      Z=1 IF INPUT RECEIVED , Z=0 OF NO HEX RECEIVED                   *
                      ;* REGISTERS ARE TRANSPARENT                                           *
                      ;*************************************************************************

                      ;* EXECUTE SINGLE OR EXTENDED ROM EXPRESSION HANDLER
                      ;*
                      ;* THE FLAG "DELIM" IS USED AS FOLLOWS:
                      ;*      DELIM = 0       NO LEADING BLANKS, NO FORCED TERMINATOR
                      ;*      DELIM = CHR     ACCEPT LEADING 'CHR'S, FORCED TERMINATOR

FCDF                  BLDNNB
FCDF  4F                  CLRA                    ;* NO DYNAMIC DELIMITER
FCE0  8C                  FCB     SKIP2           ;* SKIP NEXT INSTRUCTION
                      ;* NUILD LEADING BLANKS
FCE1                  BLDNUM
FCE1  8620                LDA     #' '            ;* ALLOW LEADING BLANKS
FCE3  978E                STA     DELIM           ;* STORE AS DELIMITER
FCE5  6E9DE303            JMP     [VECTAB+_EXPAN,PCR]     ;* TO EXP ANALYZER

                      ;* THIS IS THE DEFAULT SINGLE ROM ANALYZER. WE ACCEPT
                      ;* 1> HEX INPUT.
                      ;* 2> 'M' FOR LAST MEMORY EXAMIN ADDRESS
                      ;* 3> 'P' FOR PROGRAM COUNTER ADDRESS
                      ;* 4> 'W' FOR WINDOW VALUE
                      ;* 5> '@' FOR INDIRECT VALUE

FCE9                  EXP1
FCE9  3414                PSHS    X,B             ;* SAVE REGISTERS
FCEB                  EXPDLM
FCEB  8D5C                BSR     BLDHXI          ;* CLEAR NUMBER CHECT FORST CHARACTER
FCED  2718                BEQ     EXP2            ;* IF HEX DIGIT CONTINUE BULIDING
                      ;* SKIP BLANK IF DESIRED
FCEF  918E                CMPA    DELIM           ;* CORRECT DELIMITER
FCF1  27F8                BEQ     EXPDLM          ;* YES, IGNORE IT

                      ;* TEST FOR M OR P
FCF3  9E9E                LDX     ADDR            ;* DEFAULT FOR 'M'
FCF5  814D                CMPA    #'M'            ;* MEMORY EXAMIN ADDR WANTED ?
FCF7  2716                BEQ     EXPTDL          ;* BRANCH IF SO
FCF9  9E93                LDX     PCNTER          ;* DEFAULT FOR 'P'
FCFB  8150                CMPA    #'P'            ;* LAST PROGRAM COUNTER WANTED ?
FCFD  2710                BEQ     EXPTDL          ;* BRANCH IF SO
FCFF  9EA0                LDX     WINDOW          ;* DEFAULT TO WINDOW
FD01  8157                CMPA    #'W'            ;* WINDOW WANTED ?
FD03  270A                BEQ     EXPTDL
FD05                  EXPRTN
FD05  3594                PULS    PC,X,B          ;* RETURN AND RESTORE REGISTERS
                      ;* GOT HEX, NOW CONTINUE BULIDING
FD07                  EXP2
FD07  8D44                BSR     BLDHEX          ;* COMPUTE NEXT DIGIT
FD09  27FC                BEQ     EXP2            ;* CONTINUE IF MORE
FD0B  200A                BRA     EXPCDL          ;* SEARCH FOR +/-
                      ;* STORE VALUE AND CHECK IF NEED DELIMIER
FD0D                  EXPTDI
FD0D  AE84                LDX     ,X              ;* INDIRECTION DELIMITER
FD0F                  EXPTDL
```

```
FD0F  9F9B                STX     NUMBER          ;* STORE RESULT
FD11  0D8E                TST     DELIM           ;* TO FORCE A DELIMITER ?
FD13  27F0                BEQ     EXPRTN          ;* RETURN IF NOT WITH VALUE
FD15  8D62                BSR     READ            ;* OBTAIN NEXT CHARACTER
                    ;* TEST FOR + OR -
FD17                EXPCDL
FD17  9E9B                LDX     NUMBER          ;* LOAD LAST VALUE
FD19  812B                CMPA    #'+'            ;* ADD OPERATOR ?
FD1B  260E                BNE     EXPCHM          ;* BRANCH NOT
FD1D  8D23                BSR     EXPTRM          ;* COMPUTE NEXT TERM
FD1F  3402                PSHS    A               ;* SAVE DELIMITER
FD21  DC9B                LDD     NUMBER          ;* LOAD NEW TERM
FD23                EXPADD
FD23  308B                LEAX    D,X             ;* ADD TO X
FD25  9F9B                STX     NUMBER          ;* STORE AS NEW RESULT
FD27  3502                PULS    A               ;* RESTORE DELIMITER
FD29  20EC                BRA     EXPCDL          ;* NEW TEST IT
FD2B                EXPCHM
FD2B  812D                CMPA    #'-'            ;* SUBTRACT OPERATOR ?
FD2D  2707                BEQ     EXPSUB          ;* BRANCH IF SO
FD2F  8140                CMPA    #'@'            ;* INDIRECTION DESIRED ?
FD31  27DA                BEQ     EXPTDI          ;* BRANCH IF SO
FD33  5F                  CLRB                    ;* SET DELIMITER RETURN
FD34  20CF                BRA     EXPRTN          ;* AND RETURN TO CALLER
FD36                EXPSUB
FD36  8D0A                BSR     EXPTRM          ;* OBTAON NEXT TERM
FD38  3402                PSHS    A               ;* SAVE DELIMITER
FD3A  DC9B                LDD     NUMBER          ;* LOAD UP NEXT TERM
FD3C  40                  NEGA
FD3D  50                  NEGB
FD3E  8200                SBCA    #0              ;* CORRECT FOR A
FD40  20E1                BRA     EXPADD          ;* GO ADD TO EXPRESSION
                    ;* COMPUTE NEXT EXPRESSION TERM
                    ;* OUTPUT : X = OLD VALUE
                    ;*         'NUMBER' = NEXT TERM

FD42                EXPTRM
FD42  8D9D                BSR     BLDNUM          ;* OBTAIN NEXT BYTE
FD44  2732                BEQ     CNVRTS          ;* RTURN IF VALID NUMBER
FD46                BLDBAD
FD46  16FC13              LBRA    CMDBAD          ;* ABORT COMMAND IF INVALID

                    ;*******************************************************************************
                    ;*      BULID BINARYVALUE USING INPUT CHARACTERS.                              *
                    ;*      INPUT : A = ASCII VALUE OR DELIMITER                                   *
                    ;*              SP+0 = RETURN ADDRESS                                          *
                    ;*              SP+2 = 16 BIT RESULT AREA                                      *
                    ;*      OUTPUT: Z=1 A = BINARY VALUE                                           *
                    ;*              Z=0 IF INVALID HEX CHARACTER ( A UNCHANGED)                    *
                    ;*      VOLATILE: D                                                            *
                    ;*******************************************************************************

FD49                BLDHXI
FD49  0F9B                CLR     NUMBER          ;* CLEAR NUMBER
FD4B  0F9C                CLR     NUMBER+1        ;* CLEAR NUMBER
FD4D                BLDHEX
FD4D  8D2A                BSR     READ            ;* GET INPUT CHARACTER
FD4F                BLDHXC
FD4F  8D11                BSR     CNVHEX          ;* CONVERT AND TEST CHARACTER
FD51  2625                BNE     CNVRTS          ;* RETURN IF NOT A NUMBER
FD53  C610                LDB     #16             ;* PREPARE SHIFT
FD55  3D                  MUL                     ;* BY FOUR PLACES
FD56  8604                LDA     #4              ;* ROTATE BINARY INTO VALUE
FD58                BLDSHF
FD58  58                  ASLB                    ;* OBTAIN NEXT BIT
FD59  099C                ROL     NUMBER+1        ;* INTO LOW BYTE
FD5B  099B                ROL     NUMBER          ;* INTO HI BYTE
FD5D  4A                  DECA                    ;* COUNT DOWN
FD5E  26F8                BNE     BLDSHF          ;* BRANCH IF MORE TO DO
FD60  2014                BRA     CNVOK           ;* SET GOOD RETURN CODE

                    ;*******************************************************************************
                    ;*      CONVERT ASCII CHARACTER TO BINARY BYTE                                 *
                    ;*      INPUT : A = ASCII                                                      *
                    ;*      OUTPUT: Z = 1 A = BINARY NUMBER                                        *
                    ;*              Z = 0 IF INVALID                                               *
                    ;*      ALL REGISTERS TRANSPARENT                                              *
                    ;*      ( A UNALTERED IF INVALID HEX )                                         *
                    ;*******************************************************************************

FD62                CNVHEX
FD62  8130                CMPA    #'0'            ;* LOWER THAN A ZERO
FD64  2512                BLO     CNVRTS          ;* BRANCH NOT VALUE
FD66  8139                CMPA    #'9'            ;* POSSIBLE A-F
FD68  2F0A                BLE     CNVGOT          ;* BRANCH NOT TO ACCEPT
FD6A  8141                CMPA    #'A'            ;* LESS THAN TEN ?
FD6C  250A                BLO     CNVRTS          ;* RETURN IF MINUS ( INVALID )
FD6E  8146                CMPA    #'F'            ;* NOT TOO LARGE ?
```

```
FD70  2206                    BHI     CNVRTS          ;* NO, RETURN TOO LARGE
FD72  8007                    SUBA    #7              ;* DOWN TO BINARY
FD74                  CNVGOT
FD74  840F                    ANDA    #$0F            ;* CLEAR HIGH HEX
FD76                  CNVOK
FD76  1A04                    ORCC    #4              ;* FORCE ZERO ON FOR VALID HEX
FD78                  CNVRTS
FD78  39                      RTS                     ;* RETURN TO CALLER

                      ;* GET INPUT CHAR, ABORT COMMAND IF CONTROL-X (CANCEL)

FD79  3F              READ    SWI                     ;* GET NEXT CHARACTER
FD7A  00                      FCB     INCHNP          ;* FUNCTION
FD7B  8118                    CMPA    #CAN            ;* ABORT COMMAND ?
FD7D  27C7                    BEQ     BLDBAD          ;* BRANCH TO ABORT IF SO
FD7F  39                      RTS                     ;* RETURN TO CALLER

                      ;* G
                      ;*********************************************************************************
                      ;*                          GO - START PROGRAM EXECUTION                         *
                      ;*********************************************************************************

FD80                  CGO
FD80  8D01                    BSR     GOADDR          ;* BUILD ADDRESS IF NEEDED
FD82  3B                      RTI                     ;* START EXECUTING

                      ;* FIND OPTIONAL NEW PROGRAM COUNTER, ALSO ARM BREAKPOINTS

FD83                  GOADDR
FD83  3530                    PULS    Y,X             ;* RECOVER RETURN ADDRESS
FD85  3410                    PSHS    X               ;* STORE RETURN BACK
FD87  2619                    BNE     GONDFT          ;* IF NO CARRIAGE RETURN THEN NEW PC
                      ;* DEFAULT PROGRAM COUNTER, SO FALL THRU IF IMMEDIATE BREAKPOINT.
FD89  1701B6                  LBSR    CBKLDR          ;* SEARCH BREAKPOINTS
FD8C  AE6C                    LDX     12,S            ;* LOAD PROGRAM COUNTER
FD8E                  ARMBLP
FD8E  5A                      DECB                    ;* COUNT DOWN
FD8F  2B16                    BMI     ARMBK2          ;* DONE, NONE TO SINGLE TRACE
FD91  A630                    LDA     -NUMBKP*2,Y     ;* PRE-FETCH OPCODE
FD93  ACA1                    CMPX    ,Y++            ;* IS THIS A BREAKPOINT
FD95  26F7                    BNE     ARMBLP          ;* LOOP IF NOT
FD97  813F                    CMPA    #$3F            ;* SWI BREAKPOINTED ?
FD99  2602                    BNE     ARMNSW          ;* NO, SKIP SETTING OF PASS BRKPNT
FD9B  97FB                    STA     SWIBFL          ;* SHOW UPCOMMING SWI NOT BRKPNT
FD9D                  ARMNSW
FD9D  0C8F                    INC     MISFLG          ;* FLAG THRU A BREAKPOINT
FD9F  160106                  LBRA    CDOT            ;* DO SINGLE TRACE W/O BREAKPOINTS
                      ;* OBTAIN NEW PROGRAM COUNTER
FDA2                  GONDFT
FDA2  1700BB                  LBSR    CDNUM           ;* OBTAIN NEW PROGRAM COUNTER
FDA5  ED6C                    STD     12,S            ;* STORE INTO STACK
FDA7                  ARMBK2
FDA7  170198                  LBSR    CBKLDR          ;* OBTAIN TABLE
FDAA  00FA                    NEG     BKPTCT          ;* COMPLEMENT TO SHOW ARMED
FDAC                  ARMLOP
FDAC  5A                      DECB                    ;* DONE ?
FDAD  2BC9                    BMI     CNVRTS          ;* RETURN WHEN DONE
FDAF  A6B4                    LDA     [,Y]            ;* LOAD OPCODE
FDB1  A730                    STA     -NUMBKP*2,Y     ;* STORE INTO OPCODE TABLE
FDB3  863F                    LDA     #$3F            ;* READY "SWI" OPCODE
FDB5  A7B1                    STA     [,Y++]          ;* STTORE AND MOVE UP TABLE
FDB7  20F3                    BRA     ARMLOP          ;* AND CONTINUE

                      ;*********************************************************************************
                      ;*                          CALL - CALL ADDRESS AS SUBROUTINE                    *
                      ;*********************************************************************************

FDB9                  CCALL
FDB9  8DC8                    BSR     GOADDR          ;* FETCH ADDRESS IF NEEDED
FDBB  357F                    PULS    U,Y,X,DP,D,CC   ;* RESTORE USERS REGISTERS
FDBD  ADF1                    JSR     [,S++]          ;* CALL USER SUBROUTINE
FDBF                  CGOBRK
FDBF  3F                      SWI                     ;* PERFORM BREAKPOINT
FDC0  0A                      FCB     BRKPT           ;* FUNCTION
FDC1  20FC                    BRA     CGOBRK          ;* LOOP UNTIL USER CHANGES PC

                      ;*********************************************************************************
                      ;*                          MEMORY - DISPLAY/CHANGE MEMORY                       *
                      ;*********************************************************************************

                      ;* CMEMN AND CMPADP ARE DIRECT ENTRY POINTS FROM
                      ;* THE COMMAND HANDLER FOR QUICK COMMANDS

FDC3  17009A          CMEM    LBSR    CDNUM           ;* OBTAIN ADDRESS
FDC6  DD9E            CMEMN   STD     ADDR            ;* STORE DEFAULT
FDC8  9E9E            CMEM2   LDX     ADDR            ;* LOAD POINTER
FDCA  17FC0C                  LBSR    ZOUT2H          ;* SEND OUT HEX VALUE OF BYTE
FDCD  862D                    LDA     #'-'            ;* LOAD DELIMITER
```

```
FDCF  3F                  SWI                   ;* SEND OUT
FDD0  01                  FCB     OUTCH         ;* FUNCTION
FDD1                CMEM4
FDD1  17FF0B              LBSR    BLDNNB        ;* OBTAIN NEW BYTE VALUE
FDD4  270A                BEQ     CMENUM        ;* BRANCH IF NUMBER
                    ;* COMA - SKIP BYTE
FDD6  812C                CMPA    #','          ;* COMMA ?
FDD8  260E                BNE     CMNOTC        ;* BRANCH NOT
FDDA  9F9E                STX     ADDR          ;* UPDATE POINTER
FDDC  3001                LEAX    1,X           ;* TO NEXT BYTE
FDDE  20F1                BRA     CMEM4         ;* AND INPUT IT
FDE0                CMENUM
FDE0  D69C                LDB     NUMBER+1      ;* LOAD LOW BYTE VALUE
FDE2  8D47                BSR     MUPDAT        ;* GO OVERLAY MEMORY BYTE
FDE4  812C                CMPA    #','          ;* CONTINUE WITH NO DISPLAY ?
FDE6  27E9                BEQ     CMEM4         ;* BRANCH YES
                    ;* QUOTE STRING
FDE8                CMNOTC
FDE8  8127                CMPA    #"'"          ;* QUOTE STRING ?
FDEA  260C                BNE     CMNOTQ        ;* BRANCH NO
FDEC                CMESTR
FDEC  8D8B                BSR     READ          ;* OBTAIN NEXT CHARACTER
FDEE  8127                CMPA    #"'"          ;* END OF QUOTED STRING
FDF0  270C                BEQ     CMSPCE        ;* YES, QUIT STRING MODE
FDF2  1F89                TFR     A,B           ;* TO B FOR SUBROUTINE
FDF4  8D35                BSR     MUPDAT        ;* GO UPDATE BYTE
FDF6  20F4                BRA     CMESTR        ;* GET NEXT CHARACTER
                    ;* BLANK - NEXT BYTE
FDF8                CMNOTQ
FDF8  8120                CMPA    #$20          ;* BLANK FOR NEXT BYTE ?
FDFA  2606                BNE     CMNOTB        ;* BRANCH NOT
FDFC  9F9E                STX     ADDR          ;* UPDATE POINTER
FDFE                CMSPCE
FDFE  3F                  SWI                   ;* GIVE SPACE
FDFF  07                  FCB     SPACE         ;* FUNCTION
FE00  20C6                BRA     CMEM2         ;* NOW PROMPT FOR NEXT
                    ;* LINEFEED - NEXT BYTE WITH ADDRESS
FE02                CMNOTB
FE02  810A                CMPA    #LF           ;* LINEFEED FOR NEXT BYTE
FE04  2608                BNE     CMNOTL        ;* BRANCH NO
FE06  860D                LDA     #CR           ;* GIVE CARRIAGE RETURN
FE08  3F                  SWI                   ;* TO CONSOLE
FE09  01                  FCB     OUTCH         ;* FUNCTION
FE0A  9F9E                STX     ADDR          ;* STORE NEXT ADDRESS
FE0C  200A                BRA     CMPADP        ;* BRANCH TO SHOW
                    ;* UP ARROW - PREVIOUS BYTE AND ADDRESS
FE0E                CMNOTL
FE0E  815E                CMPA    #'^'          ;* UP ARROW FOR PREVIOUS BYTE ?
FE10  260A                BNE     CMNOTU        ;* BRANCH NOT
FE12  301E                LEAX    -2,X          ;* DOWN TO PREVIOUS BYTE
FE14  9F9E                STX     ADDR          ;* STORE NEW POINTER
FE16                CMPADS
FE16  3F                  SWI                   ;* FORCE NEW LINE
FE17  06                  FCB     PCRLF         ;* FUNCTION
FE18                CMPADP
FE18  8D07                BSR     PRTADR        ;* GO PRINT ITS VALUE
FE1A  20AC                BRA     CMEM2         ;* THEN PROMPT FOR INPUT
                    ;* SLASH - NEXT BYTE WITH ADDRESS
FE1C                CMNOTU
FE1C  812F                CMPA    #'/'          ;* SLASH FOR CURRENT DISPLAY ?
FE1E  27F6                BEQ     CMPADS        ;* YES, SEND ADDRESS
FE20  39                  RTS                   ;* RETURN FROM COMMAND

                    ;* PRINT CURRENT ADDRESS

FE21                PRTADR
FE21  9E9E                LDX     ADDR          ;*  LOAD POINTER VALUE
FE23  3410                PSHS    X             ;* SAVE X ON STACK
FE25  30E4                LEAX    ,S            ;* POINT TO IT FOR DISPLAY
FE27  3F                  SWI                   ;* DISPLAY POINTER IN HEX
FE28  05                  FCB     OUT4HS        ;* FUNCTION
FE29  3590                PULS    PC,X          ;* RECOVER POINTER AND RETURN
                    ;* UPDATE BYTE
FE2B                MUPDAT
FE2B  9E9E                LDX     ADDR          ;* LOAD NEXT BYTE POINTER
FE2D  E780                STB     ,X+           ;* STORE AND INCREMENT X
FE2F  E11F                CMPB    -1,X          ;* SUCCESSFUL STORE ?
FE31  2603                BNE     MUPBAD        ;* BRANCH FOR '?' IF NOT
FE33  9F9E                STX     ADDR          ;* STORE NEW POINTER VALUE
FE35  39                  RTS
FE36                MUPBAD
FE36  3402                PSHS    A             ;* SAVE ACCA
FE38  863F                LDA     #'?'          ;* SHOW INVALID
FE3A  3F                  SWI                   ;* SEND OUT
FE3B  01                  FCB     OUTCH         ;* FUNCTION
FE3C  3582                PULS    PC,A          ;* RETURN TO CALLER

                    ;*******************************************************************************
```

```
                        ;*                      WINDOW - SET WINDOW VALUE                              *
                        ;*****************************************************************************

FE3E                    CWINDO
FE3E  8D20                      BSR     CDNUM           ;* OBTAIN WINDOW VALUE
FE40  DDA0                      STD     WINDOW          ;* STORE IT IN
FE42  39                        RTS

                        ;*****************************************************************************
                        ;*                      DISPLAY - HIGH SPEED DISPLAY MEMORY                    *
                        ;*****************************************************************************

FE43                    CDISP
FE43  8D1B                      BSR     CDNUM           ;* FETCH ADDRESS
FE45  C4F0                      ANDB    #$F0            ;* FORCE TO 16 BOUNDARY
FE47  1F02                      TFR     D,Y             ;* SAVE IN Y
FE49  302F                      LEAX    15,Y            ;* DEFAULT LENGTH
FE4B  2504                      BCS     CDISPS          ;* BRANCH IF END OF INPUT
FE4D  8D11                      BSR     CDNUM           ;* OBTAIN COUNT
FE4F  30AB                      LEAX    D,Y             ;* ASSUME COUNT, COMPUTE END ADDR
FE51                    CDISPS
FE51  3430                      PSHS    Y,X             ;* SETUP PARAMETERS FOR HSDATA
FE53  10A362                    CMPD    2,S             ;* WAS IT COUNT ?
FE56  2302                      BLS     CDCNT           ;* BRANCH YES
FE58  EDE4                      STD     ,S              ;* STORE HIGH ADDRESS
FE5A                    CDCNT
FE5A  AD9DE184                  JSR     [VECTAB+_HSDTA,PCR]     ;* CALL PRINT ROUTINE
FE5E  35E0                      PULS    PC,U,Y          ;* CLEAN STACK AND END COMMAND

                        ;* OBTAIN NUMBER - ABORT IF NONE
                        ;* ONLY DELIMITERS OF CR, BLANK, OR '/' ARE ACCEPTED
                        ;* OUTPUT : D = VALUE, C=1 IF CARRIAGE RETURN DELIMITER, ELSE C=0

FE60                    CDNUM
FE60  17FE7E                    LBSR    BLDNUM          ;* OBTAIN NUMBER
FE63  2609                      BNE     CDBADN          ;* BRANCH IF INVALID
FE65  812F                      CMPA    #'/'            ;* VALID DELIMITER
FE67  2205                      BHI     CDBADN          ;* BRANCH IF NOT FOR ERROR
FE69  810E                      CMPA    #CR+1           ;* LEAVE COMPARE FOR CARRIAGE RET
FE6B  DC9B                      LDD     NUMBER          ;* LOAD NUMBER
FE6D  39                        RTS                     ;* RETURN WITH COMPARE
FE6E                    CDBADN
FE6E  16FAEB                    LBRA    CMDBAD          ;* RETURN TO ERROR MECHANISM

                        ;*****************************************************************************
                        ;*                      PUNCH - PUNCH MEMORY IN S1/S9 FORMAT                   *
                        ;*****************************************************************************

FE71                    CPUNCH
FE71  8DED                      BSR     CDNUM           ;* OBTAIN START ADDRESS
FE73  1F02                      TFR     D,Y             ;* SAVE IN Y
FE75  8DE9                      BSR     CDNUM           ;* OBTAIN END ADDRESS
FE77  6FE2                      CLR     ,-S             ;* SETUP PUNCH FUNCTION CODE
FE79  3426                      PSHS    Y,D             ;* STORE VALUES ON STACK
FE7B                    CCALBS
FE7B  AD9DE165                  JSR     [VECTAB+_BSON,PCR]      ;* INITIALIZE HANDLER
FE7F  AD9DE163                  JSR     [VECTAB+_BSDTA,PCR]     ;* PERFORM FUNCTION
FE83  3401                      PSHS    CC              ;* SAVE RETURN CODE
FE85  AD9DE15F                  JSR     [VECTAB+_BSOFF,PCR]     ;* TURN OFF HANDLER
FE89  3501                      PULS    CC              ;* OBTAIN CONDITION CODE SAVED
FE8B  26E1                      BNE     CDBADN          ;* BRANCH IF ERROR
FE8D  35B2                      PULS    PC,Y,X,A        ;* RETURN FROM COMMAND

                        ;*****************************************************************************
                        ;*              LOAD - LOAD MEMORY FROM S1/S9 FORMAT                           *
                        ;*****************************************************************************

FE8F                    CLOAD
FE8F  8D01                      BSR     CLVOFS          ;* CALL SETUP AND PASS CODE
FE91  01                        FCB     1               ;* LOAD FUNCTION CODE FOR PACKET
FE92                    CLVOFS
FE92  33F1                      LEAU    [,S++]          ;* LOAD CODE IN HIGH BYTE OF U
FE94  33D4                      LEAU    [,U]            ;* NOT CHANGING CC AND RESTORE S
FE96  2703                      BEQ     CLVDFT          ;* BRANCH IF CARRIAGE RETURN NEXT
FE98  8DC6                      BSR     CDNUM           ;* OBTAIN OFFSET
FE9A  8C                        FCB     SKIP2           ;* SKIP DEFAULT OFFSET
FE9B                    CLVDFT
FE9B  4F                        CLRA                    ;* CREATE ZERO OFFSET
FE9C  5F                        CLRB                    ;* AS DEFAULT
FE9D  344E                      PSHS    U,DP,D          ;* SETUP CODE, NULL WORD, OFFSET
FE9F  20DA                      BRA     CCALBS          ;* ENTER CALL TO BS ROUTINES

                        ;*****************************************************************************
                        ;*                      VERIFY - VERIFY MEMORY WITH FILES                      *
                        ;*****************************************************************************
FEA1                    CVER
FEA1  8DEF                      BSR     CLVOFS          ;* COMPUTE OFFSET IF ANY
FEA3  FF                        FCB     -1              ;* VERIFY FNCTN CODE FOR PACKET
```

```
                        ;*******************************************************************************
                        ;*              TRACE - TRACE INSTRUCTIONS                                     *
                        ;*              .     - SINGLE STEP TRACE                                      *
                        ;*******************************************************************************

FEA4                    CTRACE
FEA4  8DBA                      BSR     CDNUM           ;* OBTAIN TRACE COUNT
FEA6  DD91                      STD     TRACEC          ;* STORE COUNT
FEA8                    CDOT
FEA8  3262                      LEAS    2,S             ;* RID COMMAND RETURN FROM STACK
FEAA                    CTRCE3
FEAA  EEF80A                    LDU     [10,S]          ;* LOAD OPCODE TO EXECUTE
FEAD  DF99                      STU     LASTOP          ;* STORE FOR TRACE INTERRUPT
FEAF  DEF6                      LDU     VECTAB+_PTM     ;* LOAD PTM ADDRESS
FEB1  CC0701                    LDD     #(7 << 8)+1     ;* CYCLES DOWN , CYCLES UP
FEB4  ED42                      STD     PTMTM1-PTM,U    ;* START NMI TIMEOUT
FEB6  3B                        RTI                     ;* RETURN FOR ONE INSTRUCTION

                        ;*******************************************************************************
                        ;*              NULLS   -       SET NEW LINE AND CHAR PADDING                  *
                        ;*******************************************************************************

FEB7                    CNULLS
FEB7  8DA7                      BSR     CDNUM           ;* OBTAIN NEWLINE PAD
FEB9  DDF2                      STD     VECTAB+_PAD     ;* RESET VALUES
FEBB  39                        RTS                     ;* END COMMAND

                        ;*******************************************************************************
                        ;*              STLEVEL -       SET STACK TRACE LEVEL                          *
                        ;*******************************************************************************

FEBC                    CSTLEV
FEBC  2705                      BEQ     STLDFT          ;* TAKE DEFAULT
FEBE  8DA0                      BSR     CDNUM           ;* OBTAIN NEW STACK LEVEL
FEC0  DDF8                      STD     SLEVEL          ;* STORE NEW ENTRY
FEC2  39                        RTS                     ;* TO COMMAND HANDLER
FEC3                    STLDFT
FEC3  306E                      LEAX    14,S            ;* COMPUTE NMI COMPARE
FEC5  9FF8                      STX     SLEVEL          ;* AND STORE IT
FEC7  39                        RTS                     ;* END COMMAND

                        ;*******************************************************************************
                        ;*              OFFSET  -       COMPUTE SHORT AND LONG BRANCH OFFSETS          *
                        ;*******************************************************************************

FEC8                    COFFS
FEC8  8D96                      BSR     CDNUM           ;* OBTAIN INSTRUCTION ADDRESS
FECA  1F01                      TFR     D,X             ;* USE AS FROM ADDRESS
FECC  8D92                      BSR     CDNUM           ;* OBTAIN TO ADDRESS
                        ;* D = TO INSTRUCTION , X = FROM INSTRUCTION OFFSET BYTE(S)
FECE  3001                      LEAX    1,X             ;* ADJUST FOR *+2 SHORT BRANCH
FED0  3430                      PSHS    Y,X             ;* STORE WORK AND VALUE ON S
FED2  A3E4                      SUBD    ,S              ;* FIND OFFSET
FED4  EDE4                      STD     ,S              ;* SAVE OVER STACK
FED6  3061                      LEAX    1,S             ;* POINT FOR ONE BYTE DISPLAY
FED8  1D                        SEX                     ;* SIGN EXTEND LOW BYTE
FED9  A1E4                      CMPA    ,S              ;* VALID ONE BYTE OFFSET ?
FEDB  2602                      BNE     COFNO1          ;* BRANCH IF NOT
FEDD  3F                        SWI                     ;* SHOW ONE BYTE OFFSET
FEDE  04                        FCB     OUT2HS          ;* FUNCTION
FEDF                    COFNO1
FEDF  EEE4                      LDU     ,S              ;* RELOAD OFFSET
FEE1  335F                      LEAU    -1,U            ;* CONVERT TO LONG BRANCH OFFSET
FEE3  EF84                      STU     ,X              ;* STORE BACK WHERE X POINTS NOW
FEE5  3F                        SWI                     ;* SHOW TWO BYTE OFFSET
FEE6  05                        FCB     OUT4HS          ;* FUNCTION
FEE7  3F                        SWI                     ;* FORCE NEW LINE
FEE8  06                        FCB     PCRLF           ;* FUNCTION
FEE9  3596                      PULS    PC,X,D          ;* RESTORE STACK AND END COMMAND
                        ;*H

                        ;*******************************************************************************
                        ;*      BREAKPOINT - DISPLAY/ENTER/DELETE/CLEAR BREAKPOINTS                    *
                        ;*******************************************************************************

FEEB                    CBKPT
FEEB  2723                      BEQ     CBKDSP          ;* BRANCH DISPLAY OF JUST 'B'
FEED  17FDF1                    LBSR    BLDNUM          ;* ATTEMP VALUE ENTRY
FEF0  272C                      BEQ     CBKADD          ;* BRANCH TO ADD IF SO
FEF2  812D                      CMPA    #'-'            ;* CORRECT DELIMITER ?
FEF4  263F                      BNE     CBKERR          ;* NO, BRANCH FOR ERROR
FEF6  17FDE8                    LBSR    BLDNUM          ;* ATTEMP DELETE VALUE
FEF9  2703                      BEQ     CBKDLE          ;* GOT ONE, GO DELETE IT
FEFB  0FFA                      CLR     BKPTCT          ;* WAS 'B - ', SO ZERO COUNT
FEFD                    CBKRTS
FEFD  39                        RTS                     ;* END OF COMMAND
                        ;* DELETE THE ENTRY
```

```
FEFE  8D40        CBKDLE  BSR     CBKSET          ;* SETUP REGISTERS AND VALUE
FF00  5A          CBKDLP  DECB                    ;* ANY ENTRIES IN TABLE ?
FF01  2B32                BMI     CBKERR          ;* BRANCH NO, ERROR
FF03  ACA1                CMPX    ,Y++            ;* IS THIS THE ENTRY
FF05  26F9                BNE     CBKDLP          ;* NO, TRY NEXT
                    ;* FOUND , NOW MOVE OTHERS UP IN ITS PLACE
FF07              CBKDLM
FF07  AEA1                LDX     ,Y++            ;* LOAD NEXT ONE UP
FF09  AF3C                STX     -4,Y            ;* MOVE DOWN BY ONE
FF0B  5A                  DECB                    ;* DONE ?
FF0C  2AF9                BPL     CBKDLM          ;* NO, CONTINUE MOVE
FF0E  0AFA                DEC     BKPTCT          ;* DECREMENT BREAKPOINT COUNT
FF10              CBKDSP
FF10  8D2E                BSR     CBKSET          ;* SETUP REGISTERS AND LOAD VALUE
FF12  27E9                BEQ     CBKRTS          ;* RETURN IF NONE TO DISPLAY
FF14              CBKDSL
FF14  30A1                LEAX    ,Y++            ;* POINT TO NEXT ENTRY
FF16  3F                  SWI                     ;* DISPLAY IN HEX
FF17  05                  FCB     OUT4HS          ;* FUNCTION
FF18  5A                  DECB                    ;* COUNT DOWN
FF19  26F9                BNE     CBKDSL          ;* LOOP IF MORE TO DO
FF1B  3F                  SWI                     ;* SKIP TO NEW LINE
FF1C  06                  FCB     PCRLF           ;* FUNCTION
FF1D  39                  RTS                     ;* RETURN TO END COMMAND
                    ;* ADD NEW ENTRY
FF1E              CBKADD
FF1E  8D20                BSR     CBKSET          ;* SET UP REGISTERS
FF20  C108                CMPB    #NUMBKP         ;* ALREADY FULL ?
FF22  2711                BEQ     CBKERR          ;* BRANCH ERROR IF SO
FF24  A684                LDA     ,X              ;* LOAD BYTE TO TRAP
FF26  E784                STB     ,X              ;* TRY TO CHANGE
FF28  E184                CMPB    ,X              ;* CHANGABLE RAM ?
FF2A  2609                BNE     CBKERR          ;* BRANCH ERROR IF NOT
FF2C  A784                STA     ,X              ;* RESTORE BYTE
FF2E              CBKADL
FF2E  5A                  DECB                    ;* COUNT DOWN
FF2F  2B07                BMI     CBKADT          ;* BRANCH IF DONE TO ADD IT
FF31  ACA1                CMPX    ,Y++            ;* ENTRY ALREADY HERE ?
FF33  26F9                BNE     CBKADL          ;* LOOP IF NOT
FF35              CBKERR
FF35  16FA24              LBRA    CMDBAD          ;* RETURN TO ERROR PRODUCE
FF38              CBKADT
FF38  AFA4                STX     ,Y              ;* ADD THIS ENTRY
FF3A  6F31                CLR     -NUMBKP*2+1,Y   ;* CLEAR OPTIONAL BYTE
FF3C  0CFA                INC     BKPTCT          ;* ADD ONE TO COUNT
FF3E  20D0                BRA     CBKDSP          ;* AND NOW DISPLAY ALL OF THEM
                    ;* SETUP REGISTERS FOR SCAN
FF40              CBKSET
FF40  9E9B                LDX     NUMBER          ;* LOAD VALUE DESIRED
FF42              CBKLDR
FF42  318DE06C            LEAY    BKPTBL,PCR      ;* LOAD START OF TABLE
FF46  D6FA                LDB     BKPTCT          ;* LOAD ENTRY COUNT
FF48  39                  RTS                     ;* RETURN

                    ;*************************************************************************
                    ;*      ENCODE - ENCODE A POSTBYTE                                       *
                    ;*************************************************************************

FF49              CENCDE
FF49  6FE2                CLR     ,-S             ;* DEFAULT TO NOT INDIRECT
FF4B  5F                  CLRB                    ;* ZERO POSITIVE VALUE
FF4C  308C3F              LEAX    <CONV1,PCR      ;* START TABLE SEARCH
FF4F  3F                  SWI                     ;* OBTAIN FIRST CHARACTER
FF50  00                  FCB     INCHNP          ;* FUNCTION
FF51  815B                CMPA    #'[             ;* INDIRECT HERE ?
FF53  2606                BNE     CEN2            ;* BRANCH IF NOT
FF55  8610                LDA     #$10            ;* SET INDIRECT BIT ON
FF57  A7E4                STA     ,S              ;* SAVE FOR LATER
FF59              CENGET
FF59  3F                  SWI                     ;* OBTAIN NEXT CHARACTER
FF5A  00                  FCB     INCHNP          ;* FUNCTION
FF5B              CEN2
FF5B  810D                CMPA    #CR             ;* END OF ENTRY
FF5D  270C                BEQ     CEND1           ;* BRANCH YES
FF5F              CENLP1
FF5F  6D84                TST     ,X              ;* END OF TABLE ?
FF61  2BD2                BMI     CBKERR          ;* BRANCH ERROR IF SO
FF63  A181                CMPA    ,X++            ;* THIS THE CHARACTER
FF65  26F8                BNE     CENLP1          ;* BRANCH IF NOT
FF67  EB1F                ADDB    -1,X            ;* ADD THIS VALUE
FF69  20EE                BRA     CENGET          ;* GET NEXT INPUT
FF6B              CEND1
FF6B  308C49              LEAX    <CONV2,PCR      ;* POINT AT TABLE 2
FF6E  1F98                TFR     B,A             ;* SAVE COPY IN A
FF70  8460                ANDA    #$60            ;* ISOLATE REGISTER MASK
FF72  AAE4                ORA     ,S              ;* ADD IN INDIRECTION BIT
FF74  A7E4                STA     ,S              ;* SAVE BACK AS POSTBYTE SKELETON
FF76  C49F                ANDB    #$9F            ;* CLEAR REGISTER BIT
```

```
FF78                    CENLP2
FF78  6D84                      TST     ,X              ;* END OF TABLE ?
FF7A  27B9                      BEQ     CBKERR          ;* BRANCH ERROR IF SO
FF7C  E181                      CMPB    ,X++            ;* SAME VALUE ?
FF7E  26F8                      BNE     CENLP2          ;* LOOP IF NOT
FF80  E61F                      LDB     -1,X            ;* LOAD RESULT VALUE
FF82  EAE4                      ORB     ,S              ;* ADD TO BASE SKELETON
FF84  E7E4                      STB     ,S              ;* SAVE POSTBYTE ON STACK
FF86  30E4                      LEAX    ,S              ;* POINT TO IT
FF88  3F                        SWI                     ;* SEND OUT AS HEX
FF89  04                        FCB     OUT2HS          ;* FUNCTION
FF8A  3F                        SWI                     ;* TO NEXT LINE
FF8B  06                        FCB     PCRLF           ;* FUNCTION
FF8C  3584                      PULS    PC,B            ;* END OF COMMAND
                        ;* TABLE ONE DEFINES VALID INPUT IN SEQUENCE
FF8E                    CONV1
FF8E  4104420544064801           FCB     'A',$04,'B',$05,'D',$06,'H',$01
FF96  4801480148002C00           FCB     'H',$01,'H',$01,'H',$00,',',$00
FF9E  2D092D0153705930           FCB     '-',$09,'-',$01,'S',$70,'Y',$30
FFA6  555058102B072B01           FCB     'U',$50,'X',$10,'+',$07,'+',$01
FFAE  5080430052005D00           FCB     'P',$80,'C',$00,'R',$00,']',$00
FFB6  FF                        FCB     $FF             ;* END OF TABLE
                        ;* CONV2 USES ABOVE CONVERSION TO SET POSTBYTE BIT SKELETON.
FFB7                    CONV2
FFB7  10841100                  FDB     $1084,$1100     ;* R,           H,R
FFBB  12881389                  FDB     $1288,$1389     ;* HH,R         HHHH,R
FFBF  14861585                  FDB     $1486,$1585     ;* A,R          B,R
FFC3  168B1780                  FDB     $168B,$1780     ;* D,R          ,R+
FFC7  18811982                  FDB     $1881,$1982     ;* ,R++         ,-R
FFCB  1A83828C                  FDB     $1A83,$828C     ;* ,--R         HH,PCR
FFCF  838D039F                  FDB     $838D,$039F     ;* HHHH,PCR     [HHHH]
FFD3  00                        FCB     0               ;* END OF TABLE

                        ;*******************************************************************************
                        ;*              DEFAULT INTERRUPT TRANSFERS                                    *
                        ;*******************************************************************************

FFD4  6E9DDFEE          RSRVD   JMP     [VECTAB+_RSVD,PCR]      ;* RESERVED VECTOR
FFD8  6E9DDFEC          SWI3    JMP     [VECTAB+_SWI3,PCR]      ;* SWI3 VECTOR
FFDC  6E9DDFEA          SWI2    JMP     [VECTAB+_SWI2,PCR]      ;* SWI2 VECTOR
FFE0  6E9DDFE8          FIRQ    JMP     [VECTAB+_FIRQ,PCR]      ;* FIRQ VECTOR
FFE4  6E9DDFE6          IRQ     JMP     [VECTAB+_IRQ,PCR]       ;* IRQ VECTOR
FFE8  6E9DDFE4          SWI     JMP     [VECTAB+_SWI,PCR]       ;* SWI VECTOR
FFEC  6E9DDFE2          NMI     JMP     [VECTAB+_NMI,PCR]       ;* NMI VECTOR

                        ;*******************************************************************************
                        ;*              ASSIST09 MARDWARE VECTOR TABLE                                 *
                        ;*      THIS TABLE IS USED IF THE ASSIST09 ROM ADDRESSES THE MC6809            *
                        ;*      HARDWARE VECTORS                                                       *
                        ;*******************************************************************************

FFF0                            ORG     ROMBEG+ROMSIZ-16        ;* SETUP HARDWARE VECTORS
FFF0  FFD4                      FDB     RSRVD                   ;* RESERVED SLOT
FFF2  FFD8                      FDB     SWI3                    ;* SOFTWARE INTERRUPT 3
FFF4  FFDC                      FDB     SWI2                    ;* SOFTWARE INTERRUPT 2
FFF6  FFE0                      FDB     FIRQ                    ;* FAST INTERRUPT REQUEST
FFF8  FFE4                      FDB     IRQ                     ;* INTERRUPT REQUEST
FFFA  FFE8                      FDB     SWI                     ;* SOFTWARE INTERRUPT
FFFC  FFEC                      FDB     NMI                     ;* NON-MASKABLE INTERRUPT
FFFE  F837                      FDB     RESET                   ;* RESTART

0000                            END     RESET
```

S123F800308DE7BE1F101F8B979D3384318C35EF81C61634041F20E3A1ED816AE426F6C6E7
S123F8200DA6A0A7805A26F9318DF7D48E20FEACA12602ADA43584328DE7168DC34F1F8B18
S123F8403F0820F90158029202900028E0270028A0045022BFFE3029002840296028A029312
S123F8600290039A02B702D202BFE792047D012DE00800050000E000000000003901940143
S123F880B101CB01C30175017301C001790055017D025601D16A8DE6F7170225EE6A335F05
S123F8A00DFB261117069B505A2B0A11A3A126F8EF6A16021E0FFB3706C10B1022020FEF27
S123F8C06A58338CB8ECC56ECB4153534953543039041 0DF976D61260DAD9DE6F9AD9DE6DD
S123F8E0FB308CE53F039EF6270D6F026F03CC01A6A701E7846F013F061706462A0C50D780
S123F900FA5A2B06A630A7B120F7AE6A9F93863E3F0133E4DF954F5FDD9BDD8FDD91C60278
S123F92034071704543 08D0581812E275A308D04E9812F27528120231434026C5F812F2723
S123F9404F17040B27026A5E17042E20E8800DA75D9EC4E6802A109EEE5C27F710DE9530A0
S123F9608D015A3F0220905AE15F24033A20E4315DA65F8002A75E5AA680A1A226EE6A5EF2
S123F98026F53AEC1E308B6D5D32C4AD1E16FF7A6D5E2BC83088AEDC9B20ECFE0442054DF2
S123F9A0044304170444049D0445059F044703D2044C04DD044D040D044E04FD044F050AA3
S123F9C0045004AF04520284045304F2045404D6045604CF04570468FFA6803406C6103D5B
S123F9E08D043506840F8B90198940196E9DE5EE8DE78DE5AF648620203DA661813422399D
S123FA00109EC2EEA6EF64AF7E272EAFA6202A8D5D8D5F24FA4D27F9817F27F5A7610D8F49
S123FA202617810D2604860A8DC20DF4260BA661308C09810A270F8DB30C903B04308CFC5C
S123FA40860D8DA8860A8DA4A680810426F88D1E8D061FA9E7E420E18D18240581182602EF
S123FA6053398D0A8D0C24FA811827F44F396E9DE578AD9DE562847F394F502D048D420D8B
S123FA808F26340D902B29306C9CF82523308CE93F02098E308DE5013F058D172537068E18
S123FAA025339E91272F301F9F9127298DAA25251603F71701B9390F8F1702EB3B3F0720B8
S123FAC004E68DE4D81F9BA163272510DE97308CEC3F038DDE16FE218DE720F7DEF0A6C413
S123FAE0442402A6413986039EF0A7848651A784393447DEF08D1B81102712D6F2810D26C4
S123FB0002D6F34FE7E48C8D096AE42AFA35C717FF5CE6C4C50227F7A7413986116D6626BA
S123FB20014C3F010C8F3986143F014A3F010A8F8E61A8301F26FC39EE626D662754327DD5
S123FB403F00815326FA3F0081392722813126F26FE48D21E7618D1DE7628D19A66231CB7C
S123FB608D13270C6D692B02E7A4E1A027F235924C27CD20F98D12C6103D8D0D3404ABE057
S123FB801F89AB62A7626A63393F001701D427F835F2DEF2AE643456CC0018D7F23F01C60D
S123FBA004DDF2EC68A362108300182502C6175CE7E4CB03E761308C333F035F30618D2754
S123FBC08D258D23AE628D1F6AE426FAAF6253E76130618D14AE68AC6224C8308C113F0398
S123FBE0EC64DDF24F35D6EB8416FDED5331045339303330303030346430D0A043F06C6062D
S123FC003F075A26FB5F1F9817FDDB3F073F075CC11025F23F06252F30643F05AE64C610F1
S123FC203F045A26FB3F07AE64C610A6802B0481202402862E3F015A26F1AC622409AF6405
S123FC40A6654826CF20B53F06398D234C8D21395043FF1341000A42000B58FF0D59FF0F1A
S123FC6055FF1153FF0143430009445000 0C004F30E8103432318CD8ECA04D2F043F0120BB
S123FC80F7862D3F0130E56DE426126D3F27033F058C3F04ECA05D26DF3F0635B28D4027E1
S123FCA010810D271EE63F5A50583F075A26FB20E3A7E4DC9B6D3F2602A682ED84A6E481F8
S123FCC00D26D1308DE28AC6153502A7805A26F910EE88ECC615A68234025A26F920BC4FF2
S123FCE08C8620978E6E9DE30334148D5C2718918E27F89E9E814D27169E93815027109EEC
S123FD00A08157270A35948D4427FC200AAE849F9B0D8E27F08D629E9B812B260E8D233445
S123FD2002DC9B308B9F9B350220EC812D2707814027DA5F20CF8D0A3402DC9B40508200CC
S123FD4020E18D9D273216FC130F9B0F9C8D2A8D112625C6103D860458099C099B4A26F85B
S123FD6020148130251281392F0A8141250A814622068007840F1A04393F00811827C73920
S123FD808D013B3530341026191701B6AE6C5A2B16A630ACA126F7813F260297FB0C8F16C0
S123FDA001061700BBED6C17019800FA5A2BC9A6B4A730863FA7B120F38DC8357FADF13FCE
S123FDC00A20FC17009ADD9E9E9E17FC0C862D3F0117FF0B270A812C260E9F9E300120F1CD
S123FDE0D69C8D47812C27E98127260C8D8B8127270C1F898D3520F4812026069F9E3F0761
S123FE0020C6810A2608860D3F019F9E200A815E260A301E9F9E3F068D0720AC812F27F6F9
S123FE20399E9E341030E43F0535909E9EE780E11F26039F9E393402863F3F0135828D2007
S123FE40DDA0398D1BC4F01F02302F25048D1130AB343010A3622302EDE4AD9DE18435E037
S123FE6017FE7E2609812F2205810EDC9B3916FAEB8DED1F028DE96FE23426AD9DE165ADB2
S123FE809DE1633401AD9DE15F350126E135B28D010133F133D427038DC68C4F5F344E2087
S123FEA0DA8DEFFF8DBADD913262EEF80ADF99DEF6CC0701ED423B8DA7DDF23927058DA091
S123FEC0DDF839306E9FF8398D961F018D9230013430A3E4EDE430611DA1E426023F04EEC7
S123FEE0E4335FEF843F053F063596272317FDF1272C812D263F17FDE827030FFA398D40D7
S123FF005A2B32ACA126F9AEA1AF3C5A2AF90AFA8D2E27E930A13F055A26F93F06398D207B
S123FF20C1082711A684E784E1842609A7845A2B07ACA126F916FA24AFA46F310CFA20D053
S123FF409E9B318DE06CD6FA396FE25F308C3F3F00815B26068610A7E43F00810D270C6DD1
S123FF60842BD2A18126F8EB1F20EE308C491F988460AAE4A7E4C49F6D8427B9E18126F837
S123FF80E61FEAE4E7E430E43F043F06358441044205440648014801480148002C002D090F

```
S123FFA02D01537059305550581028072B01508043005200 5D00FF108411001288138914A8
S123FFC0861585168B1780188119821A83828C838D039F006E9DDFEE6E9DDFEC6E9DDFEAB2
S123FFE06E9DDFE86E9DDFE66E9DDFE46E9DDFE2FFD4FFD8FFDCFFE0FFE4FFE8FFECF83779
S903F837CD
```